目　录

Contents

第 1 章

导　　论

非线性代数方程组的构造性理论及有关算法，在计算机自动推理、数学机械化、技术和工程学科的智能化等领域正在发挥着越来越重要的作用．这一课题处于计算机科学、数学和工程技术的交叉位置，它的理论、算法、应用等三个方面，受到不同学科的科学家们的共同关注．如机器人智能控制就是这样的一个范例．

本书将从算法化的观点来研究非线性代数方程组（或称为多项式方程组）．这不是一个全新的方向，乃是在计算机及其创新性的应用的冲击下代数几何中消沉有年的构造性理论的复兴和发展．

§1 代 数 方 程

我们接触方程，通常是从一个只含一个未知元的一次方程这种很简单的情形开始的，然后从两个方向去发展．一方面，研究含有两个未知元的两个一次方程的方程组，以及含有三个未知元的三个一次方程的方程组；另一方面，研究一个只含一个未知元的二次方程和某些很容易化为二次方程的特殊类型的高次方程（例如四次方程）．

这两个方向的进一步发展，形成了代数学的两个最基本的部门．其中的一个称作线性代数，它是从任意的一次方程组或叫作线性方程组这个问题的研究开始的（在这里，方程的个数不一定要等于未知元的个数）．线性方程组的理论所解决的就是求解一般的线性方程组的完整方法；内容包括方程组的线性相关性、相容方程组

的求解方法 (如克莱姆 (G.Cramer) 法则、高斯 (G.F.Gauss) 消去法) 及解的结构等. 同时也形成了行列式、矩阵及向量空间等理论, 这些概念和方法, 在很多数学的研究中, 在物理和力学中, 都是重要的工具.

另一个方向的发展, 形成了所谓的多项式代数, 研究只含一个未知元但是有任意次数的方程. 由于二次方程有求解公式, 历史上关于这部分代数的兴趣主要集中于寻找高次方程的解的相应公式. 在 16 世纪找到了三次和四次方程的解的公式. 以后就开始寻找五次和更高次方程的解的公式, 即要从方程的系数出发, 通过一些根式, 可能是多层的根式来表出方程的解. 这种尝试一直继续到 19 世纪初, 最后证明了这种公式是不可能找到的, 而且对于次数等于或大于五次的方程, 都有这样的整系数的具体例子存在, 它的解是不可能利用根式来写出的. 这个事实并不值得遗憾: 与其他类型的方程 (如微分方程) 的情形一样, 不存在显式解 (或称封闭解) 一点也不影响人们对解本身性态的了解; 同时, 由于各种各样求近似解方法的发展, 也不会给实际用途带来损害. 寻找高次方程求解公式的尝试尽管失败了, 却引出了一个极其深刻的概念——群, 它对数学的影响无论怎样说都不为过分.

关于有许多个未知元, 不是线性而是任意次的方程组问题, 即非线性代数方程组的问题, 合并了前述两个方向的发展, 是代数几何学的论题. 从 19 世纪 40 年代至 20 世纪 20 年代这段时期内, 形成了的消去法理论, 这是一套用显式算法程序来解代数方程组的理论, 是一条富有成效且经得起考验的路线. 然而, 经过半个多世纪的数学公理化运动之后, 消去法理论已经从所谓的现代代数几何中“消去”了.

从几何观点看, 可以把代数方程 $f(x,y)=0$ 想象为平面上满足 $f(a,b)=0$ 的点 (a,b) 所组成的平面代数曲线. 比如方程 $x^2+y^2-1=0$, 显而易见, 就表示平面上一个圆. 一般地可把有限个 n 个变元的代数方程的公共零点 (即解) 的全体, 想象为在 n 维

空间中的一个几何构形，或"代数簇"．这种几何描述，给代数方程组的研究带来了几何的趣味和想象．正是在这个意义下，代数方程组的理论可以称为代数几何学．在存在性的代数几何里，一般要假定代数簇具有某种简单的规范性态，例如不可约性，至于如何把一般的情形化归于这种规范情形，则是不加考虑的．构造性的代数几何则不躲避这些困难；对于其中的大多数问题，如果不能以构造性方式加以解决，只能说对它们了解得还很不够．

非线性代数方程组的理论，在过去的 20 年里发生了巨大的变化：其算法性得到了显著加强，许多长期被冷落的方法，如特征集方法[34]，格罗布纳 (W. Gröbner) 基方法[7]，矩阵广义特征值法[5]以及结式消去法等[14]，唤起了广泛的关注并有了长足的进展，形成计算代数几何中内容最为丰富的部分．然而，要真正理解这一发展的必然性及其意义，必须置身于数学的现实背景中才有可能．事实上，它正是方兴未艾的数学变革风潮的产物．这场变革风潮由现代计算机及其创新性的应用所促成．

§2 结构与计算

数学中既有结构，也有计算，这种特性在它从人们观天测地的实践中刚刚萌芽的时候就具备了．在悠久的历史长河中，数学的这两个方面相互作用，共同促进了数学的成长．

测量物体长度的时候，常常是把某种长度单位逐次置放在物体上面，然后把置放的次数作为物体的长度．在这种实际的操作中，长度被看作是可以用置放来比较并且是可以线性相加的，这是一种结构；另一方面，一次一次相加计数则是一种计算．同样地，在纯粹的数学系统中，比如说在简单的整数系统中，我们不仅知道有 $3 \times 5 = 5 \times 3$，知道它们仍然是一个整数这样一种结构，而且可以根据乘法这样一种规则来计算 $3 \times 5 = 15$．

数学的这两个基本特征，对应于数学研究的方法论上，就是所

谓公理化方法和算法化 (或构造性) 方法. 公理化方法的最原始的范例是欧几里得 (Euclid) 的《几何原本》; 它所关注的是几何的逻辑结构, 即以少数的几条简单的基本原理做基础来建立几何的公理演绎体系. 这个方面的集大成者是希尔伯特 (D.Hilbert) 的名著《几何基础》, 其中引入了彻底的、严谨的公理化方法, 它为本世纪前半叶的公理化运动开辟了道路. 布尔巴基 (N.Bourbaki) 的《数学原理》对这种方法作了更为广泛的尝试.

另一方面, 算法化方法则强调具体地给出计算某一或某一类对象的方法. 古希腊数学的核心无疑是演绎推理, 但在算法方面也取得了了不起的成就, 如阿基米德 (Archimedes) 用接近于积分的方法确定了一系列复杂图形的面积和体积, 丢番图 (Diophantus) 对整数方程作了杰出的研究等. 相反, 在中国古代数学中, 虽然有一些推理, 但占统治地位的一直是算法, 发明了许多解决各种问题的著名"算术", 如用极限方法求圆周率的割圆术, 在解算术问题时使用变元的天元术, 求解方程组的大衍求一术等.

在许多情况下, 算法化方法和公理化方法常常是同样地有效的. 然而, 提供解答毕竟比纯存在性地证明有解要有意义得多. 算法性的研究不仅可以得出较为新颖, 较为深刻的见解, 而且其成果也便于应用.

算法化方法的发展, 导致了算法概念的精确化, 从而孕育了现代计算机的诞生. 图灵 (A.M.Turing) 提出了一个计算的数学模型, 称为图灵机, 它是建立在对人的计算过程的哲学分析基础上的. 图灵的基本论题 (也称为图灵 - 丘奇 (A.Church) 论题) 是

可计算性 = 图灵可计算性.

这一论题已被当做公理一样地使用着. 它不仅是计算机科学的基础, 也是整个数学的基石之一.

算法化方法的发展给数学带来了极大的繁荣, 也为数学带来了直观的素材和众多的应用.

实际上, 在任何时候, 计算工具的技术水平对数学方法本身,

都显示出重大的影响. 在本世纪前半叶人们说"数学是结构",而现在我们可以说"数学中有结构也有计算,或许更重要的是计算". 计算和结构是数学的两个不可替代的方面,但是计算之所以能够复兴到今天这样的地位,只是在现代计算机出现之后才成为可能.

§3 计算机与数学

现代计算机以两种不同的方式对数学发生影响:一方面,它使得数学比以往任何时候都更具威力,直接地影响数学中的价值观念和方法的平衡;另一方面,通过其创新性的应用,间接地改变着数学的内容和结构. 从而给数学带来了前所未有的繁荣和发展.

计算机已被普遍地用于各类数学研究. 即使从来不用计算机的数学家,也可能要投身到由于计算机的存在而产生的课题的研究中去. 在数学的各个方面,计算机提出了新课题,给证明和发现提供了新工具,引进了新的研究方法,并且在越来越多的情形中改变着证明本身.

可是,反过来我们要说"数学是计算机的灵魂". 计算机可以运算得很快,但粗糙的计算方法其复杂度可以很快地超过计算机的能力. 理论的和实际的计算之间存在着巨大的鸿沟. 对实际的计算来说,一个好的算法是至关重要的. 实际可行性问题不能容忍粗糙的算法. 正是在这个意义下,为了扩充有兴趣的计算机应用领域,要求研究一种新的数学. 数学的飞速发展使得计算机有了越来越强大的生命力.

这种新数学一般称为机械化数学. 它对实际可行性比对理论可能性更感兴趣. 实际可行性甚至已经成为衡量数学进步的一个重要准则. 在这种新数学中,算法的实际可行性往往是以牺牲它的完全性或可靠性为代价的. 对于一个问题类,如果找不到实际可行的完全算法,很多时候仅仅给出实际可行的部分算法,即对问题类的某个子类给出实际可行的完全算法也有着重要意义;有时微

不足道的出错概率可能大量节省计算所需的资源. 这些观念强烈地体现着数学发展动力的现实性的一面, 它们在传统的所谓核心数学中却是难以立足的, 是被大大忽略了的.

数学所面临的这些新的机遇和挑战, 促使数学的发展由抽象的、结构主义的道路转向具体的、构造性的、结合实际的、结合计算机的道路.

同时, 计算机作为一种强有力的信息处理的工具, 作为一个强大的数学载体, 也正在改变着数学家的工作方式. 它正在逐渐地成为数学家不可替代的可靠助手和新思想的实验场所, 它把数学家从繁复的演算中解放出来, 使之能把更多的精力注入到新的发现和创造之中.

§4 定理机器证明

在计算机应用这一广阔领域中, 使用计算机来证明和发现数学定理的尝试已有四十多年的发展历史, 现在已形成一个规模宏大的研究领域——定理机器证明. 其近年来的发展尤为迅猛, 真让人有一日千里之感.

利用计算机作辅助演算进行单独一个定理的证明, 对数学家来说早已不是创举. 在证明定理的过程中, 常常会遇到这样的情形, 其中的某些部分需要大量的计算和推导, 它们对人来说是过于厌烦甚至实际上是不可能的, 于是可以把这部分工作交给计算机去做; 也可能在证明中要决定某些步骤, 而这种决定依赖于大量的计算和推导, 或者要依据由计算机获得的结果来帮助人建立新的证明思想. 用计算机来辅助证明单独一个定理的方法最为常见, 现在已成为数学研究的一种基本方式. 这方面给人以深刻印象的著名事例是"四色定理"的证明 (这是定理机器证明领域的一件大事, 曾引发了一场关于"什么才算是一个证明"的哲学争论, 一些人对此持有保留态度; 至今还没有人给出来四色定理的手算证明).

另一种方式可以称之为系统的机器证明，就是要提出一种处理某一类问题的一般方法，这种方法虽然并不能实际地解决这一类中每个问题，但对于适用的问题却能够给出完全的判定．这个方面侧重于数理逻辑与一般推理的研究在 50 年代就已经开始了．虽然出现了一些很有价值的成就，但总的来说进展缓慢．而与之形成鲜明对比的是，结合具体的、专门知识的路线真可谓硕果累累：其中几何定理机器证明的发展特别地引人注目．

初等几何和初等代数的判定问题早在 50 年代就为塔尔斯基(A.Tarski)[32]所解决，但其复杂度远远超越了现代计算机的能力范围．具有实际意义的最初开创性工作，是由数学机械化思想的倡导者吴文俊[8],[33],[34]于 70 年代作出的，这就是著名的几何定理机器证明的所谓吴氏方法；这是第一个可以证明非平凡定理的系统的机器证明方法．其后出现的基于结式消去法以及格罗布讷基的方法，都各具特色，吸引了众多的研究者．这些方法都是建立在非线性代数方程组构造性理论基础之上的——代数几何与代数方程组毕竟是同一事物的不同称呼而已．

一个基于消去法而别具一格的进展，是几何定理机器证明的例证法[21],[22],[46]．它大体是说，可以用举例子做实验 (象在自然科学中那样) 的方法严格证明几何定理：通过近似计算能够确定准确值；归纳法与演绎法一样地有效．虽然其原理并不深奥，但把它提到哲学方法论的高度上来看，却是令人惊奇且意味深长的．

在几何定理的机器证明领域，“可读证明”[11],[47] 的出现，使其面貌焕然一新．在这里，机器能够自动产生几何定理的简短 (很容易由人来检验) 且有几何意义的证明．人们总是希望对定理能有一个直观上的、概念上的理解．若其证明过程不可能由人来检验，疑虑的阴影就不能彻底消除．所以，要求证明具有“可读性”(当然也没有理由说任何定理都必定存在“可读”的证明)．这项工作建立了 (以几何不变量为基石的) 较简短的几何算法体系．这具有一般数学方法论的意义．非欧几何定理可读证明的自动生成最近的进展，

是一个显著的例证[9],[10].

上面的论述表达了我们对这个计算机时代数学特性的一些观点. 这构成了本书的思想基础. 这个论述诚然是不完全的, 可是其中的很多内容仍已超出了本书的论题. 我们试图以此来弥补由于本书的专门性质而难免的局限性.

§5 非线性代数方程组

现在回到本书的主题: 非线性代数方程组.

对于非线性代数方程组, 类似于线性方程组的理论, 为了求解、为了研究解的结构, 首先需要研究这些方程之间的相关性, 并给出算法把这个方程组化为与之同解的较为简单的系统, 这是一个方面; 然后就简单系统来求解、来研究解的结构, 这是另一个方面.

结式消去法的原理是: 从给定的方程组构造出一具有足够多个方程的导出方程组, 然后把这个导出方程组看作为诸未知元各不同幂积的线性方程组, 从而利用已得到充分发展的、丰富的"线性方法"来研究原来的非线性方程组. 这里"结式消去法"的含义明显地比之经典意义要广泛一些. 这种方法对上述两个方面同样地有效. 比之别的方法, 结式消去法一般具有较低的复杂度和简明的表达式. 它是本书所依据的主要法则. 值得一提的是, 这里所谓的线性方法与平常处理非线性问题的局部地线性逼近的方法是截然不同的. 刚好相反, 它是整体的, 又是完全准确的.

如果给定的方程组有如下的形状:

$$TS:\left\{\begin{array}{l} f_1(u_1,\ldots,u_r;x_1)=0,\\ f_2(u_1,\ldots,u_r;x_1,x_2)=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ f_s(u_1,\ldots,u_r;x_1,x_2,\ldots,x_s)=0. \end{array}\right.$$

其中

$$f_i(u_1,\ldots,u_r;x_1,\ldots,x_i) \qquad (i=1,2,\ldots,s)$$

是某个域 K(如有理数域, 实数域, 复数域, 有理函数域等) 上关于变元 $u_1,\ldots,u_r,x_1,\ldots,x_i$ 的多项式, 并且 x_i 真正地在 f_i 中出现, 我们就说它是一个 **三角型方程组**. 对于这种方程组, 其解的结构的研究可以逐次地归结为单变元的一般 n 次方程的研究. 这样, 方程组的问题就化为一个单变元的代数方程的问题, 所以三角型方程组就是一种足够简单的系统.

一般的非线性代数方程组可表为:

$$PS:\begin{cases} p_1(u_1,\ldots,u_r;x_1,\ldots,x_s)=0,\\ p_2(u_1,\ldots,u_r;x_1,\ldots,x_s)=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ p_n(u_1,\ldots,u_r;x_1,\ldots,x_s)=0. \end{cases}$$

其中 $p_i(u_1,\ldots,u_r;x_1,\ldots,x_s)$ 是 $K[u_1,\ldots,u_r;x_1,\ldots,x_s]$ 中的多项式 $(i=1,2,\ldots,n)$. 如果这些多项式 p_i 都是关于变元 $u_1,\ldots,u_r$, $x_1,\ldots,x_s$ 的线性式, 那就是通常的线性方程组. 求解线性方程组有两种重要的方法, 一种是有显式表达的克莱姆法则, 一种是依次消去变元而把方程组三角化的所谓高斯消去法; 类似地, 通过结式消去法可以把 PS 化为一些三角型方程组:

$$TS_i. \qquad i=1,\ldots,k$$

这种三角型方程组的解分别除去相应的另外一个三角型方程组

$$TS_i' \qquad i=1,\cdots,k$$

的解以外, 合起来恰好是 GS 的解, 即

$$\mathrm{Zero}(GS)=\bigcup_{i=1}^{k}\mathrm{Zero}(TS_i\setminus TS_i').$$

这里, $\mathrm{Zero}(GS)$ 表示方程组 GS 的解集, 而 $\mathrm{Zero}(TS\setminus TS')$ 表示方程组 TS 的解集与方程组 TS' 的解集之差集. 此时, 结式消去

法对应于解线性代数方程组的高斯消去法. 对应于克莱姆法则虽然也有一些结果, 但在通往一般性结果的道路上仍有一些困难有待解决.

如上即是本书的主要技术路线.

用线性方法来解决非线性问题这个看似平常的思想, 在我们看来是结式消去法的灵魂, 但远未得到充分利用; 可以期待, 这一思想的充分发挥将能够很好地解决有关非线性代数方程组的 (无论是理论的还是实际的) 大多数难题. 明白展现以使读者容易地了解这一思想, 是本书特别希望达到的目标.

方法论大师笛卡尔 (R.Descartes) 在其著作《思维的法则》里, 提出了一个大胆的设想 (并对之作了伟大的实践, 从而创立了解析几何学):

一切问题可以化为数学问题,

一切数学问题可以化为代数问题,

一切代数问题可以化为方程求解问题.

事情当然不都可以这样"一刀切", 可仍不失其为一条值得珍藏的锦囊妙计.

非线性代数方程组是一方充满生机但荆棘丛生的沃土, 是一块蓬勃扩展而又关山重叠的疆域, 正期待着英雄们去开拓, 期待着巨人去征服.

§6 计算机代数

人们通常认为, 数学研究是由一些叫作数学家的人所从事的专业工作. 其中, 某些部分的抽象程度使得一般人望而却步. 然而"数学计算"却是各领域广大科技工作者所共享的"生产手段", 并不是数学家的"专利". 事实上, 目前许多非数学工作者所从事的数学计算, 其规模与复杂度绝不亚于专业数学家所为. 若论在计算机

上进行实际计算, 对大部分数学家而言, 可以说是刚刚起步或者甚至尚未起步. 这一现状不能说是正常的.

数学计算这个词在许多场合往往被狭隘地理解为数值计算(浮点计算) 而忽略了其另一个组成部分, 即"符号计算". 特别是不少人对当代计算技术及计算机的功能缺乏全面了解, 在他们那里数学计算几乎成了数值计算的同义语. 实际情况并非如此.

合并一个多项式的同类项, 是最简单和最常用的符号计算之一例. 这类任务不仅在中学生做练习时经常遇到. 不少漂亮而重要的研究成果是经过最后一步合并同类项化简而得. 对于项数很大的多元多项式, 若用人手来作, 这绝对是一种枯燥而易出错的工作.

一项需要较多智慧的符号计算, 是将一整系数的 (单变元或多变元) 多项式在有理数域中分解因式. 例如给了这样两个多项式:

$$f(x) = x^{26} + x^{13} + 1,$$

$$\begin{aligned} g(x,y,z) = x^3 &+ ((y+2)z + 2y + 1)x^2 \\ &+ ((y+2)z^2 + (y^2 + 2y + 1)z + 2y^2 + y)x \\ &-(y+1)z^3 + (y+1)z^2 + (y^3 + y^2)z + y^3 + y^2, \end{aligned}$$

试问它们在有理数域中是否可约, 并将其分解为不可约因子的乘积. 这可不是一项单凭谨慎和耐心就能完成的任务 (计算机给出的结果见本书第 13 页).

再举一个较有历史兴趣的例子[31]. 给了一个一般系数的五次代数方程

$$x^5 - a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5 = 0,$$

从阿贝尔 (N.H.Abel) 那里我们知道, 这个方程没有代数解, 即它的根不能用其文字系数的根式来表示. 但是契尔恩豪斯 (Tschirnhaus) 早就指出, 用一个变元替换

$$y = b_0x^4 + b_1x^3 + b_2x^2 + b_3x + b_4$$

(其中 $b_0, \cdots, b_4$ 依赖于 $a_1, \cdots, a_5$) 可将原方程转化为如下形式:

$$y^5 + y + A = 0,$$

其中 A 是 $a_1, \cdots, a_5$ 的根式表示. 尽管布灵 (Bring) 早在 200 年前就给出了计算 A 的构造性步骤, 但如果没有当代快速计算机作为工具, 要实现这样的算法是根本不可能的.

符号计算又称为符号与代数计算、符号代数、符号操作等等, 它的正式名称是"计算机代数"[15]. 而为实现计算机代数的软件包, 叫做"计算机代数系统". 在过去三十年中, 计算机代数无论在基础研究或程序设计方面, 发展都很迅速, 迄今交付使用的系统数以百计. 这些系统, 按其用途可分为两大类.

一类是特殊用途的系统. 如 60 年代发展起来的作符号积分的程序 (即求初等函数的原函数的软件). 又如用于高能物理的计算机代数系统 SCHOONSCHIP, 用于天体力学的 CAMAL, 用于广义相对论的 SHEEP, 用于代数几何的 MACAULAY, 用于交换代数的 CoCoA, 用于群论的 CAYLEY, 用于李群李代数的 DELiA, 以及用于数论的 PARI.

我们着重强调的是一般用途的系统. 限于篇幅, 这里仅能提及传播较广, 影响较大的所谓"四大系统".

MACSYMA 和 REDUCE 是 60 年代后期发展起来的两大系统. 特别前者是一个巨型系统, 拥有大量辅助软件包, 功能很强, 历史功绩卓著. 但由于它对软硬件环境设施要求过高, 只能在极少数机器上运用, 近年来已呈"门庭冷落车马稀"之势. 针对这一情况, 最近 MACSYMA 的微机版本已交付使用. REDUCE 是由特殊用途 (高能物理) 的系统逐渐补充发展而来. 它对环境设施不那么挑剔, 而且它的源程序也是公开的, 这更有利于系统的扩充和修订. 以上两个系统都是用程序语言 LISP 来实现的.

MATHEMATICA 和 MAPLE 是更为现代化的, 用 C 语言来实现的两大系统. 前者是第一个将强有力的符号计算、数值计算

与作图功能有机协调, 并适用于广泛机型的系统. 也是目前国内拥有用户最多的系统.

根据对比使用的实际经验, 我们觉得 MAPLE 最为适合我们的用途[20]. 这多半是因为它在多项式代数与编程方面用起来得心应手. 本书的大部分算例和算法均用 MAPLE 编程, 并首先在 PC 机上实现.

譬如上面分解因式的例子, 在 586/75 微机上用 MAPLE V Release 2, 不到 1 秒钟就给出结果:

$$f(x) = (x^2+x+1)(x^{24}-x^{23}+x^{21}-x^{20}+x^{18}-x^{17}+x^{15} -x^{14}+x^{12}-x^{10}+x^9-x^7+x^6-x^4+x^3-x+1),$$

$$g(x,y,z) = (x+y+z+yz+1)(x^2+y^2+z^2+xy+xz).$$

计算机代数的出现和发展, 对现代科学计算和科学推导所面临的日益艰巨繁冗的计算 (特别是符号计算) 任务, 无异于雪中送炭. 并从而推动了计算机自动推理 (机器证明) 、数学机械化、技术和工程学科的智能化等研究领域的发展.

第 2 章

消去法基础

我们先考虑只含一个变元的非线性代数方程组. 这有两个方向. 其一是方程也只有一个的情形, 此时要了解其解的性态 (如实解的个数, 解的稳定性等), 或者更一般地要确定解的位置, 这是我们最后一章要做的事; 另一方向是方程有两个或两个以上的情形, 类似于线性方程组的理论, 要研究它们的相关性及其化简. 在线性情形, 这些内容大都是熟知的, 我们试图把这些基本思想推广到一般非线性代数方程组的情形——从简单到复杂, 从特殊到一般, 正是最基本也是最有效的研究途径. 必须指出, 这些工作中的某些部分在文献中未曾出现过, 我们目前也未能把它们全都以优美的形式推广到一般情形.

机械化数学的一个重要特征是, 一个有价值的概念应有有价值的算法来支持. 本章以充分的篇幅展示了丰富多采的消去法, 它们是与单变元方程组相关性概念紧密联系的. 所以, 本章是方程组相关性概念的基础, 更是其算法的基础.

理论是灰色的, 实例是生动的. 考察实例需要洞察力和耐心. 我们来考察一个实例: 设 a, b 是参数, x 是变元, 求解方程组

$$\left\{\begin{array}{l} f_1 = x^5 - x^4 - x^3 - x^2 + x + 1 = 0, \\ f_2 = x^4 - x + a = 0, \\ f_3 = x^3 + bx - 1 = 0. \end{array}\right. \tag{7.1}$$

对于上面这个范例, 我们给出了三种不同的解法 (参见本书 §7, §9, §11) 它们各自显示了单变元方程组理论的不同模式. 如果你想到了独特的解法, 或许可以发展出一种新的模式.

§ 7 除法与整相关性

给定多项式 $g(x)$ 和 $f(x)$, 如果存在多项式 $q(x)$, 使得

$$f(x) = q(x)g(x),$$

则称多项式 $g(x)$**整除**$f(x)$. 此时我们称方程 $g(x) = 0$ 与 $f(x) = 0$ 是 **整相关的**. 有时我们也说多项式 $g(x)$ 与 $f(x)$ 是 **整相关的**.

对于任意给定的多项式 $f(x)$ 和 $g(x)$, $g(x)$ 除 $f(x)$ 的 **余式** 和 **商式** 分别记为:

$$\text{rem}(f, g, x), \quad \text{quo}(f, g, x).$$

它们满足

$$f(x) = g(x)\text{quo}(f, g, x) + \text{rem}(f, g, x).$$

且余式 $\text{rem}(f, g, x)$ 关于变元 x 的次数小于 $g(x)$ 的次数, 即

$$\deg(\text{rem}(f, g, x), x) < \deg(g, x).$$

可以用一个简单程序得到 $g(x)$ 除 $f(x)$ 的余式和商式. 依习惯, 用 $\text{lcoeff}(\cdot, x)$ 记多项式关于 x 的首项系数.

设 $r_0 := f(x)$, $r_1 := g(x)$, $\text{quo}(f, g, x) := 0$.

Step1. 令 $n := \deg(r_1, x), m := \deg(r_0, x)$, 如果 $m < n$, 则

$$\text{rem}(f, g, x) := r_0, \quad \text{quo}(f, g, x) := \text{quo}(f, g, x).$$

Step2. 如果 $m \geq n$, 令

$$\begin{aligned}
&r := r_1;\\
&r_1 := r_0 - \tfrac{\text{lcoeff}(r_0, x)}{\text{lcoeff}(r_1, x)} x^{m-n} r_1;\\
&r_0 := r;\\
&\text{quo}(f, g, x) := \text{quo}(f, g, x) + \tfrac{\text{lcoeff}(r_0, x)}{\text{lcoeff}(r_1, x)} x^{m-n};
\end{aligned}$$

返回 Step1.

整除性定理 多项式 $g(x)$ 整除多项式 $f(x)$, 当且仅当

$$\operatorname{rem}(f, g, x) = 0.$$

根据以上熟知结果, 如下给出范例 (7.1) 的第一种解法:

我们可以假定基域是有理数域. 此时 f_1 有不可约分解

$$f_1 = (x-1)(x^2+x+1)(x^2-x-1).$$

设

$$f_{11} = x-1, \quad f_{12} = x^2+x+1, \quad f_{13} = x^2-x-1.$$

这样 (7.1) 就化为

$$\begin{cases} f_{11} = 0, \\ f_2 = 0, \\ f_3 = 0. \end{cases} \quad \text{或} \quad \begin{cases} f_{12} = 0, \\ f_2 = 0, \\ f_3 = 0. \end{cases} \quad \text{或} \quad \begin{cases} f_{13} = 0, \\ f_2 = 0, \\ f_3 = 0. \end{cases} \tag{7.2}$$

现在来看第三种情形. 用 f_{13} 来除 f_2 和 f_3:

$$\begin{aligned} f_2 &= x^4 - x + a \\ &= x^2 f_{13} + x^3 + x^2 - x + a \\ &= x^2 f_{13} + x f_{13} + 2x^2 + a \\ &= x^2 f_{13} + x f_{13} + 2 f_{13} + 2x + a + 2 \\ &= (x^2 + x + 2) f_{13} + 2x + a + 2. \\ f_3 &= x^3 + bx - 1 \\ &= x f_{13} + x^2 + (b+1)x - 1 \\ &= x f_{13} + f_{13} + (b+2)x \\ &= (x+1) f_{13} + (b+2)x. \end{aligned}$$

可用商式和余式的标准记法表示为

$$\begin{aligned} &\operatorname{quo}(f_2, f_{13}, x) = x^2 + x + 2, \qquad &&\operatorname{rem}(f_2, f_{13}, x) = 2x + a + 2; \\ &\operatorname{quo}(f_3, f_{13}, x) = x + 1, &&\operatorname{rem}(f_3, f_{13}, x) = (b+2)x. \end{aligned}$$

由于 f_{13} 是不可约的, 而余式关于变元 x 的次数比 f_{13} 的低, 所以余式应恒等于 0, 即

$$\begin{aligned} \operatorname{rem}(f_2, f_{13}, x) &= 2x + a + 2 = 0; \\ \operatorname{rem}(f_3, f_{13}, x) &= (b+2)x = 0. \end{aligned}$$

反之，如果余式等于 0 且 $f_{13}=0$，则 $f_2=0$，$f_3=0$. 从而知当 $a^2+6a+4=0$，$b=-2$ 时，$2x+a+2=0$; 否则无解. 类似地可知在 (7.2) 的第一种情形下，当 $a=b=0$ 时，$x-1=0$; 否则无解. 而在第二种情形下，当 $a=b=0$ 时，$x^2+x+1=0$; 否则无解.

这种解法首先的一步是求多项式在基域 K(比如在范例中是有理数域 $\mathbf{Q}$) 上的不可约分解，这往往是比较困难的；一旦知道了不可约分解，剩下的只要用多项式的 (带余) 除法就可以了.

为了避免做除法运算时常常要出现的分式，一般用 **伪除法** 来代替除法. 对于任意给定的多项式 $f(x)$ 和 $g(x)$，$g(x)$ 除 $f(x)$ 的 **伪余式** 和 **伪商** 分别记为：

$$\mathrm{prem}(f,g,x),\quad \mathrm{pquo}(f,g,x).$$

设 $g(x)$ 关于 x 的首项系数 $\mathrm{lcoeff}(g,x)=c$，则

$$\mathrm{prem}(f,g,x)=c^k\mathrm{rem}(f,g,x),\quad \mathrm{pquo}(f,g,x)=c^k\mathrm{quo}(f,g,x).$$

其中 k 为某一非负整数. 显然有

$$c^kf(x)=g(x)\mathrm{pquo}(f,g,x)+\mathrm{prem}(f,g,x).$$

$g(x)$ 除 $f(x)$ 的伪除法可由如下程序实现：

设 $r_0:=f(x)$,$r_1:=g(x)$, $\mathrm{pquo}(f,g,x):=0$.

Step1. 令 $n:=\deg(r_1,x)$, $m:=\deg(r_0,x)$, 如果 $m<n$, 则

$$\mathrm{prem}(f,g,x):=r_0,\quad \mathrm{pquo}(f,g,x):=\mathrm{pquo}(f,g,x).$$

Step2. 如果 $m\geq n$, 令

$r:=r_1$;
$r_1:=\mathrm{lcoeff}(r_1,x)r_0-\mathrm{lcoeff}(r_0,x)x^{m-n}r_1$;
$r_0:=r$;
$\mathrm{pquo}(f,g,x):=\mathrm{lcoeff}(r_1,x)\mathrm{pquo}(f,g,x)+\mathrm{lcoeff}(r_0,x)x^{m-n}$;

返回 Step1.

§ 8 除法的显式表示

上述求两个多项式的余式及商式的算法, 可以从解线性方程组的观点来考虑. 设 $m \geq n$,

$$f(x) = a_0x^m + a_1x^{m-1} + \cdots + a_m,$$

$$g(x) = b_0x^n + b_1x^{n-1} + \cdots + b_n.$$

$$A = \begin{pmatrix} b_0 & b_1 & \cdots & b_n & & & \\ & b_0 & b_1 & \cdots & b_n & & \\ & & \cdots & \cdots & \cdots & \cdots & \\ & & & b_0 & b_1 & \cdots & b_n \\ a_0 & a_1 & \cdots & \cdots & \cdots & \cdots & a_m \end{pmatrix}.$$

又令 $x_m = x^m, x_{m-1} = x^{m-1}, \cdots, x_0 = x^0 = 1$, 考虑如下方程组:

$$\begin{pmatrix} x^{m-n}g(x) \\ x^{m-n-1}g(x) \\ \vdots \\ x^0g(x) \\ f(x) \end{pmatrix} = A \cdot \begin{pmatrix} x_m \\ x_{m-1} \\ \vdots \\ x_0 \end{pmatrix}.$$

从这 $m-n+2$ 个联立方程中消去 $m-n+1$ 个变元 $x_m, x_{m-1}, \cdots, x_n$, 可得到一个关于变元 $x_{n-1}, x_{n-2}, \cdots, x_0$ 的线性方程, 即一个其次数小于或等于 $n-1$ 的多项式. 我们可以用两种熟知的方法来做这件事.

用高斯消去法容易得如下三角型方程组:

$$\begin{pmatrix} x^{m-n}g(x) \\ x^{m-n-1}g(x) \\ \vdots \\ x^0g(x) \\ \mathrm{rem}(f,g,x) \end{pmatrix} = \begin{pmatrix} b_0 & b_1 & \cdots & b_n & & & \\ & \cdots & \cdots & \cdots & \cdots & & \\ & & b_0 & b_1 & \cdots & b_n \\ & & & c_0 & \cdots & c_{n-1} \end{pmatrix} \cdot \begin{pmatrix} x_m \\ x_{m-1} \\ \vdots \\ x_0 \end{pmatrix}.$$

这最后一个方程即为

$$\mathrm{rem}(f,g,x) = c_0x^{n-1} + c_1x^{n-2} + \cdots + c_{n-1}.$$

我们也可以用克莱姆法则求得如下显式解：

$$\operatorname{detrem}(f,g,x)=d_0x_{n-1}+d_1x_{n-2}+\cdots+d_{n-1},$$

其中 (设当 $i>n$ 时 $b_i=0$)

$$\operatorname{detrem}(f,g,x)=\begin{vmatrix} b_0 & b_1 & \cdots & \cdots & b_{m-n} & x^{m-n}g(x)\\ & b_0 & b_1 & \cdots & b_{m-n-1} & x^{m-n-1}g(x)\\ & & \ddots & \ddots & \vdots & \vdots\\ & & & b_0 & b_1 & xg(x)\\ & & & & b_0 & x^0g(x)\\ a_0 & a_1 & \cdots & \cdots & a_{m-n} & f(x)\end{vmatrix},$$

$$d_k=\begin{vmatrix} b_0 & b_1 & \cdots & \cdots & b_{m-n} & b_{m-n-1+k}\\ & b_0 & b_1 & \cdots & b_{m-n-1} & b_{m-n-k}\\ & & \ddots & \ddots & \vdots & \vdots\\ & & & b_0 & b_1 & b_{2+k}\\ & & & & b_0 & b_{1+k}\\ a_0 & a_1 & \cdots & \cdots & a_{m-n} & a_{m-n+1+k}\end{vmatrix},$$

其中 $k=0,1,\cdots,n-1$. 不同的解法当然不可能改变解本身, 比较两者即知

$$\begin{aligned}\operatorname{detrem}(f,g,x)&=b_0^{m-n+1}f(x)+Q(x)g(x)\\&=d_0x^{n-1}+d_1x^{n-2}+\cdots+d_{n-1}\\&=b_0^{m-n+1}(c_0x^{n-1}+c_1x^{n-2}+\cdots+c_{n-1})\\&=b_0^{m-n+1}\operatorname{rem}(f,g,x),\end{aligned}$$

其中 $Q(x)$ 也是可以用显式表示出来的, 且它与 $g(x)$ 除 $f(x)$ 之商 $\operatorname{quo}(f,g,x)$ 有关系：

$$Q(x)=b_0^{m-n+1}\operatorname{quo}(f,g,x).$$

自然, 在实际计算时可以利用各种"线性"方法来消元, 并不真的要分别计算那些个行列式.

两个多项式余式和商式的这种显式算法, 不仅给实际的计算提供了新的途径, 也为理论上的表示和探索带来了方便.

§9 辗 转 相 除 法

范例 (7.1) 的第二种解法是采用辗转相除把变元消去，从而判定两个方程是否有公解，并求出相应的解. 它不依赖于多项式的不可约分解.

f_1 与 f_3 辗转相除消去 x：

$$\begin{aligned}
&r_{-1} = f_1 = x^5 - x^4 - x^3 - x^2 + x + 1,\\
&r_0 = f_3 = x^3 + bx - 1,\\
&r_1 = \mathrm{rem}(r_{-1}, r_0, x) = b(x^2 + (b+1)x - 1),\\
&r_2 = \mathrm{rem}(r_0, r_1, x) = (b+2)((b+1)x - 1),\\
&r_3 = \mathrm{rem}(r_1, r_2, x) = b/(b+1)^2.
\end{aligned}$$

当出现了一个不含变元的式子时 (视参数 b 的值而定) 这个过程就终止. 最后得到的式子常称为 f_1 与 f_3 的 **(欧几里得) 结式**, 记作 $\mathrm{Eres}(f_1, f_3, x)$. 显然

$$\mathrm{Eres}(f_1, f_3, x) = \begin{cases} 0, & 若\ b = 0, -2;\\ -1, & 若\ b = -1;\\ b/(b+1)^2, & 其它. \end{cases}$$

方程 $f_1 = 0$ 和 $f_3 = 0$ 的公解即为多项式 f_1 与 f_3 的最大公因式 $\gcd(f_1, f_3, x)$ 的根. $\gcd(f_1, f_3, x)$ 就是上述辗转相除过程中最后一个不等于 0 的式子 (除去一个常数因子外):

$$\gcd(f_1, f_3, x) = \begin{cases} x^3 - 1, & 若\ b = 0;\\ x^2 - x - 1, & 若\ b = -2;\\ 1, & 其它. \end{cases}$$

令

$$g_1 = x^3 - 1, \qquad g_2 = x^2 - x - 1.$$

这样, 方程组 (7.1) 就化为

$$\begin{cases} g_1 = 0,\\ f_2 = 0. \end{cases} \quad 或 \quad \begin{cases} g_2 = 0,\\ f_2 = 0. \end{cases} \tag{9.1}$$

现在来看后一种情形. f_2 和 g_2 辗转相除消去 x:

$$
\begin{aligned}
r_{-1} &= f_2 = x^4 - x + a,\\
r_0 &= g_2 = x^2 - x - 1,\\
r_1 &= \mathrm{rem}(r_{-1}, r_0, x) = 2x + a + 2,\\
r_2 &= \mathrm{rem}(r_0, r_1, x) = a^2 + 6a + 4.
\end{aligned}
$$

从而知

$$
\gcd(f_2, g_2, x) = \begin{cases} 2x + a + 2, & \text{若 } a^2 + 6a + 4 = 0;\\ 1, & \text{其它.} \end{cases}
$$

同理可得

$$
\gcd(f_2, g_1, x) = \begin{cases} x^3 - 1, & \text{若 } a = 0;\\ 1, & \text{其它.} \end{cases}
$$

所以方程组 (9.1) 的解 (亦即方程组 (7.1) 的解) 可表示如下:

$$
\begin{cases} x^3 - 1 = 0, & \text{若 } a = b = 0;\\ 2x + a + 2 = 0, & \text{若 } a^2 + 6a + 4 = 0, b = -2;\\ \text{无解}, & \text{其它.} \end{cases}
$$

这种解法是先求出两个方程的联立解, 然后与第三个方程比较, 而求得三个方程的联立解. 它完全基于两个多项式的结式和最大公因式算法. 比之第一种解法, 看似多做了除法, 实际上避开了多项式的不可约分解, 算法单纯, 复杂度也要低一些.

对于任意给定的多项式 $f(x)$、$g(x) \in K[x]$, 两者的 (欧几里得) 结式和最大公因式分别记为:

$$
\mathrm{Eres}(f, g, x), \quad \gcd(f, g, x).
$$

存在多项式 $P(x)$、$Q(x)$、$p(x)$、$q(x) \in K[x]$, 使

$$
\begin{aligned}
P(x)f(x) + Q(x)g(x) &= \mathrm{Eres}(f, g, x) \in K,\\
p(x)f(x) + q(x)g(x) &= \gcd(f, g, x).
\end{aligned}
$$

$\gcd(f,g,x)$ 是能同时整除 $f(x)$ 和 $g(x)$ 的次数最大的多项式. 对偶地, 它也是次数最小的非零"线性式"

$$p(x)f(x)+q(x)g(x).$$

可以用一个简单程序得到 $\mathrm{Eres}(f,g,x)$ 和 $\gcd(f,g,x)$. 无妨设 $\deg(f,x)\geq\deg(g,x)$.

令 $r_{-1}:=f(x)$, $r_0:=g(x)$.

Step1. (i) 如果 $r_0=0$, 则

$$\mathrm{Eres}(f,g,x):=0,\quad \gcd(f,g,x):=r_{-1}.$$

(ii) 如果 $\deg(r_0,x)=0$, 则

$$\mathrm{Eres}(f,g,x):=r_0,\quad \gcd(f,g,x):=r_0.$$

Step2. 如果 $\deg(g,x)>0$, 令

$$\begin{aligned}&r:=r_0;\\&r_0:=\mathrm{rem}(r_{-1},r_0,x);\\&r_{-1}:=r;\end{aligned}$$

返回 Step1.

我们把在这个程序中每次循环所得到的余式 r_0 依次记作

$$r_1,\quad r_2,\quad \ldots,\quad r_k.$$

即

$$\begin{aligned}&r_{-1}=f(x),\\&r_0=g(x),\\&r_1=\mathrm{rem}(r_{-1},r_0,x),\\&r_2=\mathrm{rem}(r_0,r_1,x),\\&\cdots\cdots\cdots\cdots\\&r_k=\mathrm{Eres}(f,g,x).\end{aligned}$$

这个序列称为多项式 $f(x)$ 和 $g(x)$ 的 **余式序列**.

采用"辗转伪除法"可以避免分式运算, 这只要在上述程序中以伪除法代替除法就可以了.

§ 10 结式消去法

用辗转相除法可以从两个多项式中消去变元而得到仅含它们的系数的一个有理式, 然后可以根据这个式子是否为 0, 来判定两个多项式是否有公根. 这样的式子称为这两个多项式的结式. 同时, 辗转消元的过程也给出了两多项式的最大公因式. 辗转相除法是一个过程性的算法, 如果不经过演算, 就不能由它了解结式的性态, 而其演算过程往往是冗长繁琐的, 所以一般不能用它来求结式和最大公因式. 好在结式消去法有许多简洁优美的显式表示. 本节介绍其中形式最为简明易记的一种: 西尔维斯特 (J.J.Sylvester) 结式消去法.

给定两个多项式

$$\begin{array}{ll} f(x) = a_0x^m + a_1x^{m-1} + \cdots + a_m & \in K[x], \\ g(x) = b_0x^n + b_1x^{n-1} + \cdots + b_n & \in K[x]. \end{array} \tag{10.1}$$

记

$$M_i = \left(\begin{array}{ccccccc} a_0 & a_1 & \cdots & a_m & & & \\ & a_0 & a_1 & \cdots & a_m & & \\ & & \cdots & \cdots & \cdots & \cdots & \\ & & & a_0 & a_1 & \cdots & a_m \\ b_0 & b_1 & \cdots & b_n & & & \\ & b_0 & b_1 & \cdots & b_n & & \\ & & \cdots & \cdots & \cdots & \cdots & \\ & & & b_0 & b_1 & \cdots & b_n \end{array}\right) \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} n-i\text{ 行} \\ \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} m-i\text{ 行} \end{array}$$

M_i 是一个 $m+n-2i$ 行、$m+n-i$ 列的矩阵. 特别地 M_0 是方阵, 称为多项式 $f(x)$ 和 $g(x)$ 关于 x 的西尔维斯特**结式矩阵**, 记为 Sylvester(f,g,x). 它的行列式记为 res(f,g,x), 即

$$\mathrm{res}(f,g,x) = \det(\mathrm{Sylvester}(f,g,x)) = \det(M_0),$$

称之为多项式 $f(x)$ 和 $g(x)$ 关于 x 的 **结式**. 这样称呼是因为它有如下性质: 如果 $a_0 \neq 0$ 或者 $b_0 \neq 0$, 则 $f(x)$ 和 $g(x)$ 有公根当且仅当 $\mathrm{res}(f,g,x) = 0$ (见 §12 定理 1).

多项式 $f(x)$ 和 $g(x)$ 的最大公因式 $\gcd(f,g,x)$ 也有简明的显式表示, 它由这些矩阵 M_i 完全决定. 为了把它清楚地表示出来, 首先需要引入几个记号.

我们用 M_i^j 来表示矩阵 M_i 的前 $m+n-2i-1$ 列和第 $m+n-2i+j$ 列形成的子矩阵. 它是一个方阵, 其行列式记为 $|M_i^j|$. 特别记 $s_i=|M_i^0|$, 称为 $f(x)$ 和 $g(x)$ 的第 i 个 **主子结式**. 设 $m\geq n$, 且对于所有 $i=0,1,\ldots,n-1$, 记

$$P_{n+1}(f,g,x)=f(x),\qquad P_n(f,g,x)=g(x),$$
$$P_i(f,g,x)=\sum_{j=0}^{i}|M_i^j|x^{i-j}=|M_i^0|x^i+|M_i^1|x^{i-1}+\cdots+|M_i^i|.$$

称这个多项式序列为 $f(x)$ 和 $g(x)$ 的 **子结式多项式序列**.

定理 1 如果 $a_0\neq 0$ 或 $b_0\neq 0$, 且

$$s_0=\cdots=s_{k-1}=0,\quad s_k\neq 0,$$

则 $f(x)$ 和 $g(x)$ 的最大公因式 $\gcd(f,g,x)=P_k(f,g,x)$.

我们曾用解线性方程组的观点讨论过多项式的带余除法. 其实, 把变元 x 的不同幂积看作是不同的变元, 然后辗转消去高次幂的做法, 在辗转相除法中得到了更为充分的体现. 这是消去法的基本思想和技巧. 可以说西尔维斯特结式消去法是具有明显"线性"形式的辗转相除法, 其基本思想如下.

记 $l=m+n-1, x_l=x^l, x_{l-1}=x^{l-1},\ldots,x_0=x^0=1$. 考虑如下关于变元 x_l, x_{l-1}, …, x_0 线性方程组:

$$M_0\cdot\begin{pmatrix}x_l\\x_{l-1}\\\vdots\\x_1\\x_0\end{pmatrix}=\begin{pmatrix}x_{n-1}f(x)\\x_{n-2}f(x)\\\vdots\\f(x)\\x_{m-1}g(x)\\x_{m-2}g(x)\\\vdots\\g(x)\end{pmatrix}.$$

从这 $m+n$ 个联立方程中消去变元 $x_l, x_{l-1}, \cdots, x_n$, 并依次消去变元 $x_{n-1}, x_{n-2}, \cdots, x_1$, 可得到一列变元逐渐减少的线性方程, 即 x 的幂次逐渐减小的多项式序列: 用辗转相除消元得到的是余式序列, 用克莱姆法则消元得到的是子结式多项式序列. 特别地, 可以得到一个关于结式的重要关系:

定理 2 存在两个多项式

$$p(x) \in K[x], \quad \deg(p(x), x) < \deg(g(x), x),$$

$$q(x) \in K[x], \quad \deg(q(x), x) < \deg(f(x), x),$$

使得

$$p(x)f(x) + q(x)g(x) = \mathrm{res}(f, g, x) \in K.$$

两个多项式的余式序列的各项和它们的子结式多项式序列的各项之间的这种确定的对应关系, 导致了一个漂亮定理, 有关内容请参见本书第 6 章.

由本节的讨论, 我们可以得到如下定理:

定理 3 当 $a_0 \neq 0$ 或者 $b_0 \neq 0$ 时, 方程组

$$\{f(x) = 0, \quad g(x) = 0\}$$

等价于方程组

$$\{P_i(f, g, x) = 0: \quad i = 0, 1, \ldots, n\}.$$

尽管从形式上看, 后一方程组有更多的方程, 可是其中包含有 $f(x)$,$g(x)$ 的最大公因式, 因而是一个较简单的方程组. 值得指出的是, 如果多项式含有参数, 那么在上述定理中 $f(x)$, $g(x)$ 的子结式多项式序列是不能由它们的余式序列来代替的. 这是因为在这种情况下不能用一个统一的公式来写出余式序列, 但子结式多项式序列却总是可用一致的式子来表示. 这是子结式多项式序列的一个特点, 在很多情况下将表现出其优越性.

§ 11 联 合 消 去 法

三个或者三个以上的多项式系统，一般不能简单地由一个"结式"来判定它们是否有公根，但可用所谓"结式组"来判定. 例如，给定三个一般的二次多项式系统：

$$\begin{cases} h_1 = a_0x^2 + a_1x + a_2, \\ h_2 = b_0x^2 + b_1x + b_2, \\ h_3 = c_0x^2 + c_1x + c_2. \end{cases}$$

容易知道，如果 $b_0 \neq 0$ 并且 $c_0 \neq 0$ 或 $a_0 \neq 0$，那么 h_1 、h_2 、h_3 有公根当且仅当如下三式都等于 0:

$$\begin{vmatrix} a_0 & a_1 & a_2 & 0 \\ 0 & a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 & 0 \\ 0 & b_0 & b_1 & b_2 \end{vmatrix}, \quad \begin{vmatrix} a_0 & a_1 & a_2 & 0 \\ 0 & a_0 & a_1 & a_2 \\ c_0 & c_1 & c_2 & 0 \\ 0 & c_0 & c_1 & c_2 \end{vmatrix}, \quad \begin{vmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{vmatrix}.$$

当然这种结式组并不是唯一的，例如可以引入一个新变元 λ，且令 $h = h_2 + \lambda h_3$，那么 $res(h_1, h, x)$ 作为 λ 的二次多项式其三个系数显然是一结式组. 明显地还可以有其他的组合. 它们在一定意义下是等价的. 一个比较好的说法是，如果 $a_0 \neq 0$ 或 $b_0 \neq 0$ 或 $c_0 \neq 0$，那么 h_1、h_2、h_3 有公根当且仅当矩阵

$$\begin{pmatrix} a_0 & a_1 & a_2 & 0 \\ 0 & a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 & 0 \\ 0 & b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 & 0 \\ 0 & c_0 & c_1 & c_2 \end{pmatrix}$$

的秩小于 4.

上面这段话具有普遍的意义. 在给出一般的方法之前，让我们先来看一个具体的例子. 我们要给出范例 (7.1) 的一个结式系统，

从而给出求解范例的第三种方法.

$$\begin{cases} f_1 = x^5 - x^4 - x^3 - x^2 + x + 1 = 0, \\ f_2 = x^4 - x + a = 0, \\ f_3 = x^3 + bx - 1 = 0. \end{cases}$$

现在要利用联立性质, 而不只局限于两个方程两个方程地消元. 这不外乎用变元的一些幂积来乘诸方程然后对它们进行线性组合.

由于我们已经对"线性化"方法有了相当的了解, 无妨直接考虑"线性"方程组 (把 x 的不同幂积 $x^7, x^6, x^5, x^4, x^3, x^2, x, 1$ 看作不同的变元):

$$\begin{pmatrix} 1 & 0 & b & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & b & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & b & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & b & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & b & -1 \\ 0 & 1 & 0 & 0 & -1 & a & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & a & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & a \\ 1 & -1 & -1 & -1 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & -1 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & -1 & -1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x^7 \\ x^6 \\ x^5 \\ x^4 \\ x^3 \\ x^2 \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

经过消元 (等价于对其系数矩阵通过行变换而三角化) 知, 它等价于方程组:



$$
\begin{pmatrix}
1 & 0 & b & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & b & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & b & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & b & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & b & -1 \\
0 & 0 & 0 & 0 & 0 & b & b+b^2 & -b \\
0 & 0 & 0 & 0 & 0 & 0 & a+b^2 & -b \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & a^2+b^2-ab+ab^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2b^2+b^3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\cdot
\begin{pmatrix}
x^7 \\ x^6 \\ x^5 \\ x^4 \\ x^3 \\ x^2 \\ x \\ 1
\end{pmatrix}
=
\begin{pmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{pmatrix}.
$$

现在明显地只需考虑最后五个非零方程, 即

$$
\begin{cases}
x^3 + bx - 1 & = 0, \\
b(x^2 + (b+1)x - 1) & = 0, \\
(a + b^2)x - b & = 0, \\
a^2 - ab + b^2 + ab^2 & = 0, \\
b^2(b+2) & = 0.
\end{cases}
$$

于是原方程组的解 (亦即方程组 (7.1) 的解) 可表示如下:

$$
\begin{cases}
x^3 - 1 = 0, & \text{若 } a = b = 0; \\
(a+4)x + 2 = 0, & \text{若 } a^2 + 6a + 4 = 0, b = -2; \\
\text{无解}, & \text{其它}.
\end{cases}
$$

一般地, 给定一组多项式:

$$
\{f_i(x) = a_{i0}x^{n_i} + a_{i1}x^{n_i-1} + \ldots + a_{in_i},\ i = 0, \ldots, k\}.
$$

不妨设

$$
n_0 \leq n_1 \leq \cdots \leq n_k.
$$

如果 f_0 和 f_k 的首项系数 a_{00} , a_{k0} 不都等于 0, 那么方程组

$$
\{f_0(x) = 0, f_1(x) = 0, \ldots, f_k(x) = 0\}
$$

等价于"线性方程组":

$$\{x^{n_k-i}f_0(x)=0, x^{n_0-1}f_j(x)=0, i=1,\dots,n_k; j=1,\dots,k\}.$$

依惯例, 这里是把变元 x 的幂积 $1, x, x^2, \dots, x^{n_0+n_k-1}$ 当做不同的变元. 这个等价性是有丰富内涵的, 比如通过适当消元可以求得诸多项式的最大公因式, 亦即原方程组的解.

自然, 前述引入新变元的方法仍然能行得通, 这里从略.

§ 12 结式的某些性质

本节推导有关结式的一些重要关系和结论. 尽管这些都是周知的, 可是这里的推导过程却具有普遍的价值, 这就是我们之所以把它写出来的原因.

多项式 $f(x)$ 和 $g(x)$ 如前. 记

$$l=m+n-1,\quad x_l=x^l,\quad x_{l-1}=x^{l-1},\quad \dots,\quad x_0=x^0=1.$$

考虑如下关于变元 $x_l, x_{l-1}, \dots, x_0$ 的线性方程组:

$$\left.\begin{aligned}
a_0x_l+a_1x_{l-1}+\cdots+a_mx_{n-1}&=0,\\
a_0x_{l-1}+a_1x_{l-2}+\cdots+a_mx_{n-2}&=0,\\
&\vdots\\
a_0x_m+a_1x_{m-1}+\cdots+a_mx_0&=0,
\end{aligned}\right\} n\text{ 个}$$

$$\left.\begin{aligned}
b_0x_l+b_1x_{l-1}+\cdots+(b_n-g)x_{m-1}&=0,\\
b_0x_{l-1}+b_1x_{l-2}+\cdots+(b_n-g)x_{m-2}&=0,\\
&\vdots\\
b_0x_n+b_1x_{n-1}+\cdots+(b_n-g)x_0&=0.
\end{aligned}\right\} m\text{ 个}$$

它的前 n 个方程仅对于多项式 $f(x)$ 的 m 个根

$$\alpha_1,\quad \alpha_2,\quad \dots,\quad \alpha_m$$

成立 (当 $a_0\neq 0$ 时, 恰好是 m 个根), 而后 m 个方程是恒等式. 因此这个方程组当且仅当 x 为多项式 $f(x)$ 的根时有解. 并且由于有

$x_0 = 1$, 这些解自然都是非平凡解. 这意味着其系数矩阵

$$M = \begin{pmatrix} a_0 & a_1 & \cdots & a_m & & & \\ & a_0 & a_1 & \cdots & a_m & & \\ & & \ddots & \ddots & & \ddots & \\ & & & a_0 & a_1 & \cdots & a_m \\ b_0 & b_1 & \cdots & b_n - g & & & \\ & b_0 & b_1 & \cdots & b_n - g & & \\ & & \ddots & \ddots & & \ddots & \\ & & & b_0 & b_1 & \cdots & b_n - g \end{pmatrix}$$

的行列式 (展开为 g 的多项式) 等于 0:

$$\det(M) = c_0 g^m + c_1 g_{m-1} + \cdots + c_m = 0.$$

这就是说, $\det(M)$ 作为 g 的 m 次多项式, 它的 m 个根为

$$g(\alpha_1),\ g(\alpha_2),\ \ldots,\ g(\alpha_m).$$

在 $\det(M)$ 的上述展开式中, 显然

$$c_0 = (-1)^m a_0^n, \quad c_m = \operatorname{res}(f, g, x).$$

由根与系数的关系, 得:

$$\prod_{i=1}^{m} g(\alpha_i) = (-1)^m \frac{c_m}{c_0}.$$

这样就得到一个重要关系

$$\operatorname{res}(f, g, x) = a_0^n \prod_{i=1}^{m} g(\alpha_i). \tag{12.1}$$

设多项式 $g(x)$ 的 n 个根是 $\beta_1, \beta_2, \ldots, \beta_n$. 根据对称性及 (12.1) 式, 可得

$$\operatorname{res}(f, g, x) = (-1)^{mn} b_0^m \prod_{j=1}^{n} f(\beta_j). \tag{12.2}$$

$$\operatorname{res}(f,g,x)=a_0^n b_0^m \prod_{i=1}^{n}\prod_{j=1}^{m}(\beta_i-\alpha_j). \tag{12.3}$$

$$\operatorname{res}(f_1 f_2,g,x)=\operatorname{res}(f_1,g,x)\operatorname{res}(f_2,g,x). \tag{12.4}$$

$$\operatorname{res}(f,g,x)=a_0^{n-d}\operatorname{res}(f,\operatorname{rem}(g,f,x),x). \tag{12.5}$$

其中 $d=\deg(\operatorname{rem}(g,f,x),x)$.

由 (12.1) 式和 (12.2) 式, 易知:

定理 1 如果 $a_0\neq 0$ 或者 $b_0\neq 0$, 则 $f(x)$ 和 $g(x)$ 有公根的充分必要条件是 $\operatorname{res}(f,g,x)=0$.

设 λ 是一个参变元, 结式 $\operatorname{res}(f,g+\lambda,x)$ 称为 $f(x)$ 与 $g(x)$ 的 λ **结式**. 它是关于 λ 的一个多项式. 我们有:

定理 2 $f(x)$ 的 m 个根中恰好有 k 个使得 $g(x)=0$ 的充要条件是结式 $\operatorname{res}(f,g+\lambda,x)$ 关于 λ 的最低次数为 k.

§ 13 用低阶行列式表示的结式

设多项式 $f(x)$ 和 $g(x)$ 的次数分别为 m 和 n, 且 $m\geq n$, $f(x)$ 和 $g(x)$ 的西尔维斯特结式是阶数为 $m+n$ 的行列式. 一般来说, 计算高阶行列式是相当困难的, 往往希望通过降阶, 来降低复杂度. 本节介绍两个降阶计算结式的方法: 贝佐 (E.Bezout) 结式消去法和友阵方法. 贝佐结式消去法所要计算的行列式的阶数是 m; 友阵方法则可以是 m, 也可以是 n.

§ 13.1 贝佐结式消去法

设

$$\begin{aligned} f(x) &= a_0x^m + a_1x^{m-1}+\cdots+a_m \quad &\in K[x],\\ g(x) &= b_0x^n + b_1x^{n-1}+\cdots+b_n \quad &\in K[x]. \end{aligned}$$

首先来看 $m=n$ 时的情形. 把 $f(x)$ 和 $g(x)$ 写为如下形式:

$$\begin{aligned} f(x) &= f_{i1}(x)x^i + f_{i2}(x),\\ g(x) &= g_{i1}(x)x^i + g_{i2}(x). \end{aligned}$$

其中

$$f_{i1}(x)=a_0x^{n-i}+\cdots+a_{n-i},\quad f_{i2}(x)=a_{n-i+1}x^{i-1}+\cdots+a_n,$$

$$g_{i1}(x)=b_0x^{n-i}+\cdots+b_{n-i},\quad g_{i2}(x)=b_{n-i+1}x^{i-1}+\cdots+b_n.$$

可以证明 $f(x)$ 和 $g(x)$ 的公根必定是多项式

$$p_i(x)=\begin{vmatrix} f_{i1}(x) & f_{i2}(x) \\ g_{i1}(x) & g_{i2}(x) \end{vmatrix}=\sum_{j=1}^{n} d_{ij}x^{n-j} \tag{13.1}$$

的根. 这是因为

$$\begin{vmatrix} f_{i1}(x) & f_{i2}(x) \\ g_{i1}(x) & g_{i2}(x) \end{vmatrix}=\begin{vmatrix} f_{i1}(x) & f_{i1}(x)x^i+f_{i2}(x) \\ g_{i1}(x) & g_{i1}(x)x^i+g_{i2}(x) \end{vmatrix}=\begin{vmatrix} f_{i1}(x) & f(x) \\ g_{i1}(x) & g(x) \end{vmatrix}.$$

考虑形如 (13.1) 的所有多项式:

$$p_{n-i+1}(x)=\sum_{j=1}^{n} d_{ij}x^{n-j}.\qquad i=1,2,\ldots,n$$

这 n 个多项式的系数矩阵

$$\begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}$$

称为多项式 $f(x)$ 和 $g(x)$ 的 **贝佐矩阵** [18], 记作 Bezout(f,g,x). 其行列式就是 $f(x)$ 和 $g(x)$ 的结式.

可以直接用多项式 $f(x)$ 和 $g(x)$ 的系数写出它们的贝佐矩阵. 记

$$[i,j]=a_ib_j-a_jb_i,$$

$$c_{ij}=\sum_{k=0}^{\min(i,j)}[k,i+j+1-k].$$

则 (因 $d_{i-1,j-1}=c_{ij}$)

$$\mathrm{Bezout}(f,g,x)=(c_{ij}),\quad i=0,1,\cdots,n-1;j=0,1,\cdots,n-1.$$

即 Bezout$(f,g,x)=$

$$\begin{pmatrix} [0,1] & [0,2] & [0,3] & \cdots & [0,n] \\ [0,2] & [0,3]+[1,2] & [0,4]+[1,3] & \cdots & [1,n] \\ [0,3] & [0,4]+[1,3] & [0,5]+[1,4]+[2,3] & \cdots & [2,n] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [0,n] & [1,n] & [2,n] & \cdots & [n-1,n] \end{pmatrix}.$$

当 $m>n$ 时, 考虑多项式 $f(x)$ 和 $x^{m-n}g(x)$. 这两个多项式的次数相等. 用上述方法可以构造出 n 个多项式:

$$p_i(x)=\sum_{j=i}^{m}c_{ij}x^{m-j}, \qquad i=m-n,\ldots,m \quad (13.2)$$

并且不用另外的 $m-n$ 个多项式

$$p_k(x)=f_{k2}(x)g(x), \qquad k=1,\ldots,m-n-1$$

而用如下 $m-n$ 个多项式

$$q_k(x)=x^kg(x). \qquad k=0,1,\ldots,m-n-1$$

代替它和 (13.2) 中的多项式一起组成 $m+n$ 个多项式. 这 $m+n$ 个多项式的系数矩阵称为多项式 $f(x)$ 和 $g(x)$ 的 **贝佐矩阵**, 记作 Bezout(f,g,x). 其行列式就是 $f(x)$ 和 $g(x)$ 的结式 (参见本书第 6 章).

§ 13.2 友阵方法

设 $f(x)$ 是首项系数为 1 的多项式:

$$f(x)=x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n\in K[x].$$

n 阶弗罗贝尼乌斯 (F.G.Frobenius) 阵：

$$F = \begin{pmatrix} 0 & & & & -a_n \\ 1 & 0 & & & -a_{n-1} \\ & 1 & \ddots & & \vdots \\ & & \ddots & 0 & -a_2 \\ & & & 1 & -a_1 \end{pmatrix}$$

称为多项式 $f(x)$ 的 **友阵**[30]．用 I_n 表示 n 阶单位阵，由行列式的计算可知：

$$|xI_n - F| = f(x).$$

这就是说，任何多项式都是其友阵的特征多项式．

多项式的根与其友阵的特征值是一一对应的．这一联系应能使我们把多项式之间的关系转化为两者友阵之间的关系．但这方面似乎仍未得到充分展开；下面的论断是简单的：

定理 1 设 $f(x)$ 和 $g(x)$ 都是首项系数为 1 的多项式，F 是 $f(x)$ 的友阵，则 $f(x)$ 和 $g(x)$ 互素，当且仅当 $g(F)$ 非异．

证明 不妨设 $m = \deg(g, x)$，$g(x)$ 的 m 个根是 $\alpha_1, \alpha_2, \cdots, \alpha_m$．因此

$$g(F) = \prod_{i=1}^{m} (F - \alpha_i I_n).$$

两边取行列式，得

$$|g(F)| = \prod_{i=1}^{m} |(F - \alpha_i I_n)|.$$

因 F 是 $f(x)$ 的友阵，所以

$$|F - \alpha_i I_n| = (-1)^n |\alpha_i I_n - F)| = (-1)^n f(\alpha_i). \quad (i = 1, 2, \ldots, m)$$

于是

$$|g(F)| = (-1)^{mn} \prod_{i=1}^{m} f(\alpha_i).$$

这样，定理就是显然的了.

在利用上述论断消元以判断两个多项式有无公根时，主要的一步是计算友阵 F 的幂. 当幂指数 k 不超过友阵 F 的阶数 n 时，F^k 有如下一个简易可行的求法——先求出 F^k 的第 1 列，再由第 1 列推出第 2 列，第 2 列推出第 3 列，等等，直至推出第 n 列.

将 F^k 按列分块为

$$F^k = (\beta_1, \beta_2, \cdots, \beta_n).$$

不难证明

$$\beta_i = F\beta_{i-1}. \qquad (i = 2,3,\ldots,n) \quad (13.3)$$

事实上，因 F 是弗罗贝尼乌斯阵，故对 $j = 2,3,\ldots,n$，有

$$e_i = Fe_{i-1};$$

其中 e_i 表示第 i 个分量为 1 其余分量都为 0 的 n 维列向量. 于是

$$\beta_i = F^k e_i = F^k(Fe_{i-1}) = F(F^k e_{i-1}) = F\beta_{i-1}.$$

(13.3) 式给出了 F^k 中后一列与前一列的关系. 又，

$$\beta_1 = F^k e_1 = F^{k-1}(Fe_1) = F^{k-1}e_2 = \cdots$$

$$= \begin{cases} Fe_k = e_{k+1}, & \text{当 } k < n \text{ 时;} \\ Fe_n = \alpha, & \text{当 } k = n \text{ 时.} \end{cases} \qquad (13.4)$$

其中 α 是 F 的末列向量. (13.4) 式给出了 F^k 的第一列.

§ 14 方程组与消去法

最后让我们回顾一下本章所给出的诸算法与方程组的关系.

在线性代数方程组的理论中，如果一个方程可以表示为其它方程的线性组合，那么该方程可以从方程组中划去而不影响方程组的解，并且说原方程组是线性相关的. 这个概念对应于非线性

代数方程组理论中，则是所谓整相关性概念. 方程 $f(x)=0$ 的解与多项式 $f(x)$ 的根 (或零点) 显然是一回事，所以有时不提方程 $f(x)=0$ 而只简单地说多项式 $f(x)$.

给定单变元方程组：

$$S_0: \quad \{f_i(x)=0: \quad i=0,1,\ldots,n\}.$$

其中 $f_i(x)\in K[x]$. 如果存在

$$q_i(x)\in K[x], \qquad\qquad i=0,1,\ldots,n$$

且其中至少有一个 $q_i(x)=1$, 使

$$\sum_{i=0}^{n} q_i(x)f_i(x)=q_0(x)f_0(x)+q_1(x)f_1(x)+\cdots+q_n(x)f_n(x)=0,$$

就说方程组 S_0 或其诸定义多项式 $f_i(x)(i=0,1,\ldots,n)$ 是 **整相关的**. 比如说 $q_0(x)=1$, 此时方程组 S_0 与方程组

$$S_1: \quad \{f_i(x)=0: \quad i=1,\ldots,n\}.$$

同解 (重解依其重数计). 所以，整相关的方程组可以通过简单地划去某个方程进行化简. 在这种情况下，

$$f_0(x)\in (f_1(x),\ldots,f_n(x)).$$

这里 $(f_1(x),\ldots,f_n(x))$ 即为 $K[x]$ 中由 $f_1(x),\ldots,f_n(x)$ 生成的 **理想**. 也记为

$$f_0\equiv 0\,(f_1,\ldots,f_n).$$

并说 $(f_1,\ldots,f_n)$ 整除 f_0.

两个单变元多项式的整相关性 (即整除性) 问题，可以用多项式的带余除法 (或伪除法) 来解决. 多个单变元多项式的整相关性问题，虽然可以用“联合消去法”的某种简单变化来解决，但一般都是两个两个地来处理.

线性代数方程组的另一种重要情形是不相容方程组 (或矛盾方程组). 这在非线性代数方程组理论中也是重要的, 对应着一个概念：方程组 (或者多项式) 的互素性.

给定单变元方程组：

$$\{f_i(x)=0: \quad i=0,1,\ldots,n\},$$

其中 $f_i(x)\in K[x]$. 如果存在

$$q_i(x)\in K[x], \qquad\qquad i=0,1,\ldots,n$$

使

$$\sum_{i=0}^{n} q_i(x)f_i(x)=q_0(x)f_0(x)+q_1(x)f_1(x)+\cdots+q_n(x)f_n(x)=1,$$

就说方程组 S_0 或其诸定义多项式 $f_i(x)(i=0,1,\ldots,n)$ 是 **不相容的**, 或 **互素的**.

两个单变元多项式是否互素, 可以用它们的结式是否非零来判定. 多个单变元多项式的互素性问题, 一般也是两个两个地来处理.

介于整相关性与互素性之间的情形, 在线性代数方程组中是不存在的；而在非线性方程组, 就是相关性概念所表达的情形.

给定单变元方程组：

$$\{f_i(x)=0: \quad i=0,1,\ldots,n\},$$

其中 $f_i(x)\in K[x]$. 无妨假定

$$m=\deg(f_0,x)\geq\deg(f_1,x)\geq\cdots\geq\deg(f_n,x)=l.$$

如果存在不都等于零的多项式

$$q_i(x)\in K[x], \qquad\qquad i=0,1,\ldots,n$$

且

$$\deg(q_n,x)<m, \quad \deg(q_i,x)<l, \qquad\qquad i=0,1,\ldots n-1$$

使

$$\sum_{i=0}^{n} q_i(x)f_i(x) = q_0(x)f_0(x) + q_1(x)f_1(x) + \cdots + q_n(x)f_n(x) = 0,$$

就说方程组 S_0 或其诸定义多项式 $f_i(x)(i = 0, 1, \ldots, n)$ 是 **相关的**.

两个单变元多项式 $f(x)$ 和 $g(x)$ 如果既不整相关又不互素, 那么它们必定有非平凡的公因子, 可以用结式消去法把它们的最大公因子 $d(x)$ 求出来. 在处理包含 $f(x) = 0$ 和 $g(x) = 0$ 的方程组时, 可以把这两个方程从原方程组中划去而代之以 $d(x) = 0$. 容易证明, 三个或者三个以上单变元的多项式总是相关的.

值得指出, 本章的重点是"子结式多项式序列", 它在理论上整齐统一, 而在算法上简洁明快. 希望读者对此特别注意 (参看第 6 章).

第 3 章

三角型方程组

对于一般的非线性代数方程组，我们首先要把它化为一些三角型方程组，然后通过对这些三角型方程组的进一步分析来了解原方程组解的性态．本章研究三角型方程组与其他方程的相关性问题，主要是前一章有关概念以及相应算法的一种推广．

为了行文方便，一般我们总把三角型方程组写为

$$TS: \begin{cases} f_1(x_1) = 0, \\ f_2(x_1, x_2) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_s(x_1, x_2, \ldots, x_s) = 0. \end{cases}$$

这时，所论基域是 K 上 $u_1, \ldots, u_r$ 的有理函数域 $K(u_1, \ldots, u_r)$，即

$$f_i \in K(u_1, \ldots, u_r)[x_1, \ldots, x_i]. \qquad i = 1, 2, \ldots, s$$

简单地，也把 $K(u_1, \ldots, u_r)$ 写为 K．我们将讨论 TS 与另一个多项式方程 $g = g(x_1, \ldots, x_s) = 0$ 的互素性、整相关性、相关性以及 TS 关于 g 的相对单纯分解等等．这是一个具有独立意义的重要问题，比如一个实际的近似计算过程总可以用一个三角型方程组来表达，该三角型方程组与另一个多项式方程的关系问题，反映着这个近似计算的精度．当然并非已无用武之地，在这里还有另外一些问题仍未得到很好解决，它们或可作为初学者入门的恰当题目．

为了对这些内容有一个直观的了解，让我们从一个例子谈起．

§ 15 一 个 例 子

求解方程组:

$$\begin{cases} f_1(x,y) = -xy^2 + 2x^2 - x - 2y + 2 = 0, \\ f_2(x,y) = x^2 - xy + y^2 - 3y + 2 = 0, \\ f_3(x,y) = x^2 + 5xy + 3y^2 - 6x + 2y - 5 = 0. \end{cases}$$

把 f_1, f_2 看作为 y 的多项式 (x 作为参数), 它们的子结式多项式序列 (其定义见 §10) 是:

$$\begin{aligned} &P_3(f_1,f_2,y) = f_1, \\ &P_2(f_1,f_2,y) = f_2, \\ &P_1(f_1,f_2,y) = (x+2)((x+1)y - (x^2+1)), \\ &P_0(f_1,f_2,y) = \mathrm{res}(f_1,f_2,y) = x^2(x-1)^2(x+2). \end{aligned}$$

由于 f_2 作为 y 的多项式其首项系数是 1, 所以 $\{f_1 = 0, f_2 = 0\}$ 的解集是如下两个方程组解集之并:

$$\{g_1(x) = x+2 = 0, \quad P_0 = 0\}, \quad \{g_2(x) = x^2(x-1)^2 = 0, \quad P_1 = 0\}.$$

这是因为, 当 $g_1(x) = 0$ 时, P_1 的首项系数 $(x+2)(x+1) = 0$,

$$\gcd(f_1, f_2, y) = P_0(f_1, f_2, y),$$

而当 $g_2(x) = 0$ 时, P_1 的首项系数 $(x+2)(x+1) \neq 0$,

$$\gcd(f_1, f_2, y) = P_1(f_1, f_2, y).$$

这样, 原方程组就化为如下两个方程组:

$$\begin{cases} g_1(x) = x + 2 = 0, \\ h_1(x,y) = y^2 - (x+3)y + x^2 + 2 = 0, \\ f_3(x,y) = x^2 + 5xy + 3y^2 - 6x + 2y - 5 = 0. \end{cases}$$

$$\begin{cases} g_2(x) = x^2(x-1)^2 = 0, \\ h_2(x,y) = (x+2)((x+1)y - (x^2+1)) = 0, \\ f_3(x,y) = x^2 + 5xy + 3y^2 - 6x + 2y - 5 = 0. \end{cases}$$

接下来需要做的, 是判定三角型方程组 $\{g_1(x)=0, h_1(x,y)=0\}$ 和 $\{g_2(x)=0, h_2(x,y)=0\}$ 各有哪些解使得方程 $f_3(x,y)=0$ 成立? 超出这个实例, 可以考虑一个略微一般的问题, 即给定三角型方程组:

$$TS: \begin{cases} f_1(x_1)=0, \\ f_2(x_1,x_2)=0, \\ \cdots\cdots\cdots\cdots \\ f_s(x_1,x_2,\ldots,x_s)=0, \end{cases}$$

及另一多项式方程

$$g(x_1,x_2,\ldots,x_s)=0,$$

试问方程组 TS 的哪些解使得方程 $g(x_1,x_2,\ldots,x_s)=0$ 成立? 此时无疑可以借用上一章的诸方法来进一步求解. 可是我们不简单套用单变元的方法, 因为那样做往往比较繁琐. 本章将给出一个系统化的方法, 下面就本节的这个简单例子来展示该方法的梗概.

首先来看方程组 $\{g_1(x)=0, h_1(x,y)=0\}$ 与方程 $f_3(x,y)=0$ 的关系. 令

$$\begin{aligned} r_1(x) &= \mathrm{res}(f_3,h_1,y) \\ &= 52x^4+82x^3-35x^2-99x, \\ r_2 &= \mathrm{res}(r_1,g_1,x)=234. \end{aligned}$$

容易知道

$$\begin{cases} g_1(x)=0, \\ h_1(x,y)=0, \\ f_3(x,y)=0 \end{cases} \Longrightarrow \quad \begin{cases} g_1(x)=0, \\ r_1(x)=0 \end{cases} \Longrightarrow \quad r_2=0.$$

而 $r_2=234$, 因此方程组 $\{g_1(x)=0, h_1(x,y)=0\}$ 与方程 $f_3(x,y)=0$ 是不相容的, 即方程组 $\{g_1(x)=0, h_1(x,y)=0, f_3(x,y)=0\}$ 无解. 在此情况下, 我们也说多项式 $f_3(x,y)$ 与三角列 $\{g_1(x), h_1(x,y)\}$ **互素**.

再来看方程组 $\{g_2(x)=0, h_2(x,y)=0\}$ 与方程 $f_3(x,y)=0$ 的关系. 令

$$
\begin{aligned}
R_1(x) &= \mathrm{res}(f_3, h_2, y)\\
&= 9x^6+39x^5+45x^4-9x^3-48x^2-36x,\\
R_2 &= \mathrm{res}(R_1, g_2, x)=0.
\end{aligned}
$$

易知多项式 $R_1(x)$ 与 $g_2(x)$ 的最大公因子是 $g(x)=x(x-1)$ (这样就有分解 $g_2(x)=(g(x))^2$), 所以只要考虑方程组 $\{g(x)=0, h_2(x,y)=0\}$ 与方程 $f_3(x,y)=0$ 的关系. 令

$$
\begin{aligned}
p_1(x) &= \mathrm{prem}(f_3, h_2, y)\\
&= 9x^6+39x^5+45x^4-9x^3-48x^2-36x,\\
p_2 &= \mathrm{prem}(p_1, g, x)=0.
\end{aligned}
$$

而

$$
\left\{\begin{array}{l} p_2=0,\\ g(x)=0,\\ h_2(x,y)=0 \end{array}\right. \Longrightarrow \left\{\begin{array}{l} p_1(x)=0,\\ g(x)=0,\\ h_2(x,y)=0 \end{array}\right. \Longrightarrow \left\{\begin{array}{l} g(x)=0,\\ h_2(x,y)=0,\\ f_3(x,y)=0. \end{array}\right.
$$

因此方程组 $\{g(x)=0, h_2(x,y)=0\}$ 与方程 $f_3(x,y)=0$ 是完全相容的, 即方程组 $\{g(x)=0, h_2(x,y)=0, f_3(x,y)=0\}$ 与方程组 $\{g(x)=0, h_2(x,y)=0\}$ 同解. 我们称多项式 $f_3(x,y)$ 与三角列 $\{g_2(x), h_2(x,y)\}$ 是 **相关** 的, 它与三角列 $\{g(x), h_2(x,y)\}$ 是 **整相关的**.

至此即知, 原方程组的解 (包括解的重数) 可表示为:

$$
\{x(x-1)=0, (x+2)((x+1)y-(x^2+1))=0\}.
$$

一个多项式与一个三角列的 **相关性**、**互素性** 以及 **整相关性** 在非线性代数方程组的求解理论中具有基本的重要性. 这些概念及其相应算法是本章的主要内容.

§ 16 互　素　性

给定一个三角型方程组：

$$TS: \begin{cases} f_1(x_1) = 0, \\ f_2(x_1, x_2) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_s(x_1, x_2, \ldots, x_s) = 0. \end{cases}$$

我们用 TS 来记其等式左边的多项式列 $\{f_1、f_2、\ldots、f_s\}$，并称之为一个 **三角列**. 对于一个多项式 $g = g(x_1, x_2, \ldots, x_s)$，如果存在多项式 $q_i \in K[x_1, x_2, \ldots, x_s], i = 0, 1, 2, \ldots, s$，使

$$q_0 g + q_1 f_1 + q_2 f_2 + \cdots + q_s f_s = 1,$$

则称多项式 g 与三角列 TS 是 **互素的**，记为

$$(g, f_1, f_2, \ldots, f_s) = (1).$$

这里 $(g, f_1, f_2, \ldots, f_s)$ 表示 $K[x_1, x_2, \ldots, x_s]$ 中所有可用形式

$$q_0 g + q_1 f_1 + q_2 f_2 + \cdots + q_s f_s$$

表示的多项式之集．即 $K[x_1, x_2, \ldots, x_s]$ 中由多项式 g，f_1，f_2，$\ldots$，f_s 所生成的 **理想**.

如果多项式 g 与三角列 TS 是互素的，那么

$$\{g = 0, f_1 = 0, f_2 = 0, \ldots, f_s = 0\} \Longrightarrow 1 = 0.$$

即方程组 $\{g = 0, f_1 = 0, f_2 = 0, \ldots, f_s = 0\}$ 是不相容的．互素性概念是非线性代数方程组理论中最基本的概念之一．

由上一章知道，两个单变元多项式的互素性可用它们的结式来判别．一个自然的想法是把结式的概念推广于多项式 g 与三角列 TS 的情形，以作为多项式 g 与三角列 TS 互素性的判定算法．这正如我们在 §15 里判定方程 $f_3 = 0$ 与三角型方程组 $\{g_1(x) = 0, h_1(x, y) = 0\}$ 不相容时所做的那样．

设多项式 g 与三角列 TS 如上, 记

$$
\begin{aligned}
& r_s = g, \\
& r_{s-1} = \mathrm{res}(r_s, f_s, x_s), \\
& r_{s-2} = \mathrm{res}(r_{s-1}, f_{s-1}, x_{s-1}), \\
& \cdots\cdots\cdots\cdots \\
& r_0 = \mathrm{res}(r_1, f_1, x_1).
\end{aligned}
$$

我们称 r_0 为多项式 g 关于三角列 TS 的 **结式**. 记为

$$\mathrm{res}(g, f_s, f_{s-1}, \ldots, f_1).$$

依惯例, 用 $\mathrm{lcoeff}(\cdot, x)$ 表示多项式关于 x 的首项系数. 记

$$c_i = \mathrm{lcoeff}(f_i, x_i). \qquad i = 1, 2, \ldots, k$$

常常把 c_i 叫做 f_i 的 **初式**. 很明显, $c_1 \in K$, $c_2 \in K[x_1]$, $\ldots$, $c_s \in K[x_1, \ldots, x_{s-1}]$. 设多项式 g 与三角列 TS 如上, 由上述归纳定义可得一列等式:

$$
\begin{aligned}
& r_s = g, \\
& r_{s-1} = a_s r_s + b_s f_s, \\
& r_{s-2} = a_{s-1} r_{s-1} + b_{s-1} f_{s-1}, \\
& \cdots\cdots\cdots\cdots \\
& r_0 = a_1 r_1 + b_1 f_1.
\end{aligned}
$$

把 $r_s, r_{s-1}, \ldots, r_0$ 由上往下顺次代入, 就得到恒等式:

$$r_0 = h_0 g + h_1 f_1 + h_2 f_2 + \cdots + h_s f_s. \tag{16.1}$$

其中 $h_1 = b_1$, $h_k = b_k \prod_{i=1}^{k-1} a_i (k = 2, \ldots, s)$, $h_0 = \prod_{i=1}^{s} a_i$. 并且归纳地可以证明: 如果 $c_i \neq 0, i = 1, 2, \ldots, s$, 那么 g 与 TS 互素, 当且仅当 $\mathrm{res}(g, f_s, f_{s-1}, \ldots, f_1) \neq 0$.

在上面这个论断中, 条件是 $c_i \neq 0 (i = 1, 2, \ldots, k)$. 然而对于一般的三角列来说, 这样的条件会带来一些问题. 比如我们取

$$
\begin{aligned}
& f_1 = x(x+2) = 0, \\
& f_2 = xy^2 + 2y - (x^2 + 1) = 0.
\end{aligned}
$$

如果要求 $\text{lcoeff}(f_2, y) = x \neq 0$, 就失去了一些有用的情形. 为了避免发生这种现象, 可以把 $\{f_1, f_2\}$ 分解为如下两组:

$$\begin{cases} f_{11} = x = 0, \\ f_{21} = 2y - (x^2+1) = 0; \end{cases} \qquad \begin{cases} f_{12} = x + 2 = 0, \\ f_{22} = xy^2 + 2y - (x^2+1) = 0. \end{cases}$$

从后文将知道, 这种分解并不难做到. 所以一般只考虑这样的三角列, 使其 $c_1 \neq 0$, c_2 与三角列 $\{f_1\}$互素, c_3与三角列$\{f_1, f_2\}$互素, $\cdots$, c_s 与三角列 $\{f_1, f_2, \ldots, f_{s-1}\}$ 互素. 递推地利用上述论断可知, 此时

$$\begin{aligned} &c_1 \neq 0, \\ &\text{res}(c_2, f_1) \neq 0, \\ &\text{res}(c_3, f_2, f_1) \neq 0, \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ &\text{res}(c_s, f_{s-1}, \ldots, f_1) \neq 0. \end{aligned}$$

这样的三角列称之为 **正常升列**[42], 或 **真升列**[50], 记为 AC.

这样就得到了如下重要的 **互素性定理**:

定理 1 多项式 $g = g(x_1, x_2, \ldots, x_s)$ 与正常升列 AC: $\{f_1, f_2, \ldots, f_s\}$ 互素, 当且仅当 $\text{res}(g, f_s, f_{s-1}, \ldots, f_1) \neq 0$.

§ 17 整 相 关 性

给定一个三角列 TS : $\{f_1, f_2, \cdots, f_s\}$, 和一个多项式 $g = g(x_1, x_2, \ldots, x_s)$, 如果存在多项式 $q_i \in K[x_1, x_2, \ldots, x_s]$, $i = 1, 2, \ldots, s$, 使 $g = q_1 f_1 + q_2 f_2 + \cdots + q_s f_s$, 即

$$g \in (f_1, f_2, \ldots, f_s),$$

则称多项式 g 与三角列 TS 是 **整相关** 的, 也说理想 $(f_1, f_2, \ldots, f_s)$ 整除多项式 g, 记为

$$g \equiv 0\,(f_1, f_2, \ldots, f_s).$$

如果 g 与三角列 TS 是整相关的, 那么

$$\{f_1 = 0, f_2 = 0, \ldots, f_s = 0\} \Longleftrightarrow \{f_1 = 0, f_2 = 0, \ldots, f_s = 0, g = 0\}.$$

在这里，整相关性概念所起的作用类似于线性方程组理论中线性相关性概念所起的作用．整相关性概念是非线性代数方程组理论中又一个最基本的概念．

要给出一个切实可行的算法来判断一个多项式与一个三角列是否整相关，远不是一件轻而易举的事．正如在单变元的情形那样，我们首先想到的当然是除法．我们要把这种运算递归地定义于一个多项式与一个三角列的情形．在这种情况下，伪除法比除法更为适用．这个思想已经在§15里出现了．在那里，我们断定方程 $f_3=0$ 与三角型方程组 $\{g_2(x)=0, h_2(x,y)=0\}$ 是"完全相容的"，这和整相关性概念是一致的．

设多项式 g 与三角列 TS 如上，记

$$\begin{aligned}
&r_s=g,\\
&r_{s-1}=\operatorname{prem}(r_s,f_s,x_s),\\
&r_{s-2}=\operatorname{prem}(r_{s-1},f_{s-1},x_{s-1}),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\\
&r_0=\operatorname{prem}(r_1,f_1,x_1).
\end{aligned}$$

我们称 r_0 为三角列 TS 除多项式 g 的**余式**，记为

$$\operatorname{prem}(g,f_s,f_{s-1},\ldots,f_1).$$

设 $c_i=\operatorname{lcoeff}(f_i,x_i)$, $i=1,2,\ldots,k$. 即 c_i 为 f_i 的初式．于是，根据定义，我们可以得到一列等式：

$$\left\{\begin{aligned}
&r_s=g,\\
&r_{s-1}=c_s^{d_s}r_s+q_sf_s,\\
&r_{s-2}=c_{s-1}^{d_{s-1}}r_{s-1}+q_{s-1}f_{s-1},\\
&\cdots\cdots\cdots\cdots\cdots\cdots\\
&r_0=c_1^{d_1}r_1-q_1f_1.
\end{aligned}\right. \tag{17.1}$$

把这些 $r_s, r_{s-1}, \ldots, r_0$ 由上往下顺次代入，就得到如下的所谓**余式公式**:

$$\left(\prod_{i=1}^{s}c_i^{d_i}\right)g=\sum_{i=1}^{s}Q_if_i+r_0. \tag{17.2}$$

其中 d_i 为非负整数, $i=1,2,\ldots,s$, 且

$$Q_1=-q_1,$$
$$Q_k=-q_k\prod_{j=1}^{k-1}c_j^{d_j}. \qquad k=2,\ldots,s$$

上述由 g 开始逐步求伪余式最后得到 r_0 的算法, 称为 **逐次伪除法**.

余式的一个显著特征是:

$$\deg(r_0,x_i)<\deg(f_i,x_i). \qquad i=1,2,\ldots,s$$

即余式中 x_i 的次数小于 f_i 中 x_i 的次数. 反过来, 如果 g 中 x_i 的次数小于 f_i 中 x_i 的次数 $(i=1,2,\ldots,s)$, 那么 $\{f_1,f_2,\ldots,f_s\}$ 除 g 的余式就是 g 本身.

引入逐次伪除法, 是用以判定多项式 g 与三角列 $TS:\{f_1,f_2,\ldots,f_s\}$ 的整相关性的. 可是对于一般的三角列来说, 这并不总是有效的. 例如, $TS_1:\{f_1=x_1^2,f_2=x_1x_2\}$, $g_1=x_1+x_2$, 在有理多项式环 $\mathbf{Q}[x_1,x_2]$ 中, 理想 (x_1^2,x_1x_2) 由所有被 x_1 整除且不含一次项的多项式组成, 显然 $g_1\notin(f_1,f_2)$, 但 $\mathrm{prem}(g_1,f_2,f_1)=0$. 又例如, TS_2: $\{h_1=x_1^2+x_1,h_2=x_1x_2^2+x_2\}$, $g_2=(x_1+1)x_2$, $g_2=(x_1+1)h_2-x_2^2h_1\in(h_1,h_2)$, 但 $\mathrm{prem}(g_2,h_2,h_1)=g_2\neq0$.

由此可见, 不受任何限制的三角列对于整相关性来说不是一个好的规范概念. 幸运的是 §16 所引入的正常升列的概念, 不仅对于互素性而且对于整相关性都是恰好的, 这反映于如下 **整相关性定理** 之中:

定理 1 多项式 $g=g(x_1,x_2,\ldots,x_s)$ 与正常升列 AC: $\{f_1,f_2,\ldots,f_s\}$ 整相关, 即

$$g\in(f_1,f_2,\ldots,f_s),$$

当且仅当

$$\mathrm{prem}(g,f_s,f_{s-1},\ldots,f_1)=0.$$

§ 18 整相关性定理的证明

为了证明整相关性定理，引入正常升列的另一种等价定义是方便的：给定三角列 $TS:\{f_1, f_2, \ldots, f_s\}$，设 c_i 是 f_i 的初式，则 TS 是 **正常升列** 当且仅当 $c_1 \neq 0$ 并且对于每个 $i = 2, \ldots, s$，初式 c_i 不是多项式环

$$K[x_1, \ldots, x_{i-1}]/(f_1, \ldots, f_{i-1})$$

中的 **零** 或 **零因子**.

这种说法不是算法性的，但在叙述中有它的便利之处. 现在让我们通过下例来直观地理解这些概念. 考虑三角列

$$TS: \quad \begin{cases} f_1 = x_1^2 - u^2, \\ f_2 = (x_1 - u)x_2^2 + x_1x_2. \end{cases}$$

把 f_1 和 f_2 看作是 u 的有理函数域 $K(u)$ 上的多项式，此时 $c_1 = 1$，$c_2 = x_1 - u$；当 $f_1 = x_1^2 - u^2 = 0$ 时有两种可能情形：

$$x_1 - u = 0, \qquad \text{或者} \qquad x_1 + u = 0.$$

即 $(x_1-u)(x_1+u) = f_1 \equiv 0(f_1)$. 所以 $c_2 = x_1 - u$ 是环 $K(u)[x_1]/(f_1)$ 中的零因子. 这说明 TS 不是正常升列.

以下证明整相关性定理.

充分性 AC: $\{f_1, f_2, \ldots, f_s\}$ 是正常升列，如果

$$\text{prem}(g, f_s, f_{s-1}, \ldots, f_1) = 0,$$

要证 g 与 AC 整相关. 考虑 (1) 中的等式，此时

$$r_0 = 0 \quad \Longrightarrow \quad c_1^{d_1} r_1 \equiv 0\,(f_1) \quad \Longrightarrow \quad r_1 \equiv 0\,(f_1).$$

这后一步成立是因为 c_1 不是环 $K[x_1]/(f_1)$ 中的零因子. 类似地，

$$r_1 \equiv 0\,(f_1) \quad \Longrightarrow \quad c_2^{d_2} r_2 \equiv 0\,(f_1, f_2) \quad \Longrightarrow \quad r_2 \equiv 0\,(f_1, f_2).$$

后一步成立是因为 c_2 不是环 $K[x_1]/(f_1,f_2)$ 中的零因子. 以此类推, 最后我们就得到

$$g = r_s \equiv 0\,(f_1, f_2, \ldots, f_s),$$

即 g 与正常升列 AC 整相关.

必要性 如果 $g = r_s \equiv 0(f_1, f_2, \ldots, f_s)$, 即存在多项式 q_1, q_2, …, $q_s \in K[x_1, x_2, \ldots, x_s]$, 使

$$g = q_1 f_1 + q_2 f_2 + \cdots + q_s f_s, \tag{18.1}$$

要证 $r_0 = \mathrm{prem}(g, f_s, f_{s-1}, \ldots, f_1) = 0$.

设 $r_0 = \mathrm{prem}(g, f_s, f_{s-1}, \ldots, f_1)$, 则根据余式公式, 此时有多项式 c, Q_1, Q_2, …, $Q_s \in K[x_1, x_2, \ldots, x_s]$, 使

$$c \cdot g = Q_1 f_1 + Q_2 f_2 + \cdots + Q_s f_s + r_0, \tag{18.2}$$

且

$$\deg(r_0, x_i) < \deg(f_i, x_i). \qquad i = 1, 2, \ldots, s \tag{18.3}$$

以 c 乘 (18.1) 式两边, 并与 (18.2) 式联立, 消去左边的 $c \cdot g$, 可得

$$r_0 = a_1 f_1 + \cdots + a_{s-1} f_{s-1} + a_s f_s, \tag{18.4}$$

其中 $a_i = cq_i - Q_i$, $i = 1, \ldots, s-1, n$. 设 $m = \deg(f_s, x_s)$, a_s 按 x_s 的降幂排列为 $a_s = b_p x_s^p + \cdots + b_1 x_s + b_0$. 把 a_i 中关于 x_s 的次数等于 $m+p$ 的项记为 $d_i x_s^{m+p}$, $i = 1, \ldots, s-1$. 设 c_s 是 f_s 的初式, 则由于 f_1, f_2, …, f_{s-1} 中没有 x_s, 利用 (18.3), 可得

$$d_1 f_1 + \cdots + d_{s-1} f_{s-1} + b_p c_s = 0.$$

于是

$$b_p c_s \equiv 0\,(f_1, \ldots, f_{s-1}).$$

因 AC: $\{f_1, f_2, \ldots, f_s\}$ 是正常升列, c_s 不是

$$K[x_1, \ldots, x_{s-1}]/(f_1, \ldots, f_{s-1})$$

中的零因子, 所以 $b_p \equiv 0\,(f_1, \ldots, f_{s-1})$. 即有多项式 $h_1, \ldots, h_{s-1}$, 使

$$b_p = h_1 f_1 + \cdots + h_{s-1} f_{s-1}.$$

把它代入 (18.4), 即得

$$r_0 = a_1^* f_1 + \cdots + a_{s-1}^* f_{s-1} + a_s^* f_s, \tag{18.5}$$

其中 $a_i^* = a_i + h_i x_s^p f_s$, $i = 1, \ldots, s-1$, $a_s^* = b_{p-1} x_s^{p-1} + \cdots + b_0$. 把以上对 (18.4) 的讨论施之于 (18.5), 即可得 $b_{p-1} \equiv 0\,(f_1, \ldots, f_{s-1})$, 即有多项式 $h_1^*, \ldots, h_{s-1}^*$, 使

$$b_{p-1} = h_1^* f_1 + \cdots + h_{s-1}^* f_{s-1}.$$

如此继续下去, 最后可得

$$r_0 = t_1 f_1 + \cdots + t_{s-1} f_{s-1} \equiv 0\,(f_1, \ldots, f_{s-1}),$$

其中 $t_i \in K(x_s)[x_1, \ldots, x_{s-1}]$.

现在把 $K(x_s)$ 当作基域 (以代替 K), 并用 r_0 代替 g, 令 $r_0^* = \mathrm{prem}(r_0, f_{s-1}, \ldots, f_1)$, 则重复以上推理, 可得

$$r_0 = r_0^* = t_1^* f_1 + \cdots + t_{s-2}^* f_{s-2} \equiv 0\,(f_1, \ldots, f_{s-2}).$$

如此继续下去, 最后就得到 $r_0 = 0$. 证毕.

§ 19 相 关 性

给定一个三角型方程组:

$$TS: \begin{cases} f_1(x_1) = 0, \\ f_2(x_1, x_2) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_s(x_1, x_2, \ldots, x_s) = 0 \end{cases}$$

和一个多项式方程 $g = g(x_1, x_2, \ldots, x_s) = 0$, 设若 TS 对应的三角列是正常升列

$$AC: \quad \{f_1, f_2, \ldots, f_s\},$$

那么方程 $g=0$ 与方程组 TS 之间的关系, 除了互素性和整相关性这两种极端情形之外, 还有怎样的中间情形呢? 这就是本节所要研究的相关性.

先引入一些记号. 令

$$n_i=\deg(f_i,x_i), \qquad i=1,2,\ldots,s$$

则 x_1 有 n_1 个解

$$\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_{n_1}.$$

对应于每一个解 $x_1=\alpha_i$, x_2 有 n_2 个解

$$\alpha_{(i,1)},\ \alpha_{(i,2)},\ \ldots,\ \alpha_{(i,n_2)}.$$

等等. 一般地, 对应于每一组

$$x_1=\alpha_{i_1},\ x_2=\alpha_{(i_1,i_2)},\ \ldots,\ x_k=\alpha_{(i_1,i_2,\ldots,i_k)},$$

x_{k+1} 有 n_{k+1} 个解

$$\alpha_{(i_1,i_2,\ldots,i_k,1)},\alpha_{(i_1,i_2,\ldots,i_k,2)},\ldots,\alpha_{(i_1,i_2,\ldots,i_k,n_{k+1})}.\quad k=1,2,\ldots,s-1$$

TS 的每一组解显然可以表示为

$$\begin{cases} x_1=\alpha_{i_1}, \\ x_2=\alpha_{(i_1,i_2)}, \\ \cdots\cdots\cdots\cdots\cdots \\ x_s=\alpha_{(i_1,i_2,\ldots,i_s)}. \end{cases}$$

这样的一组解我们简单地把它记为 $x_{(i_1,i_2,\ldots,i_s)}$, 即

$$x_{(i_1,i_2,\ldots,i_s)}=(\alpha_{i_1},\alpha_{(i_1,i_2)},\ldots,\alpha_{(i_1,i_2,\ldots,i_s)}).$$

因此, TS 的所有 $n_1n_2\cdots n_s$ 组解 (重解按其重数计) 可表示为

$$\{x_{(i_1,i_2,\ldots,i_s)}:i_1=1,\ldots,n_1;i_2=1,\ldots,n_2;\ldots;i_s=1,\ldots,n_s\}.$$

设 c_i 是 f_i 的初式 $(i=1,2,\ldots,s)$, $\nu_s=\deg(g,x_s)$,

$$\nu_j=\deg(\mathrm{res}(g,f_s,\ldots,f_{j+1}),x_j). \qquad j=1,\ldots,s-1$$

令

$$C = c_1^{\nu_1}\prod_{j=2}^{s}\prod_{i_1=1}^{n_1}\prod_{i_2=1}^{n_2}\cdots\prod_{i_{j-1}=1}^{n_{j-1}}(c_j(x_{(i_1,i_2,\ldots,i_s)}))^{\nu_j}.$$

根据第二章有关结式的性质及 $\mathrm{res}(g, f_s, \ldots, f_2, f_1)$ 的递推定义, 容易得到

$$\mathrm{res}(g, f_s, \ldots, f_2, f_1) = C\prod_{i_1=1}^{n_1}\prod_{i_2=1}^{n_2}\cdots\prod_{i_s=1}^{n_s} g(x_{(i_1,i_2,\ldots,i_s)}). \qquad (19.1)$$

设 λ 是一个参变元, 结式 $\mathrm{res}(g+\lambda, f_s, \ldots, f_2, f_1)$ 称为 g 与 $AC:\{f_1, f_2, \ldots, f_s\}$ 的 **λ 结式**, 它是关于 λ 的一个多项式. 我们有如下所谓 **相关性判准**[48],[41].

定理 1 TS 的所有 $n_1n_2\cdots n_s$ 组解

$$\{x_{(i_1,i_2,\ldots,i_s)}: i_1 = 1,\ldots,n_1; i_2 = 1,\ldots,n_2;\ldots; i_s = 1,\ldots,n_s\}.$$

中恰好有 k 个使得 $g(x) = 0$ 的充要条件是结式

$$\mathrm{res}(g+\lambda, f_s, \ldots, f_2, f_1)$$

关于 λ 的最低次数为 k.

特别地, g 与 AC 互素当且仅当 $\mathrm{res}(g+\lambda, f_s, \ldots, f_2, f_1)$ 的常数项不等于 0; g 与 AC 整相关则 $\mathrm{res}(g+\lambda, f_s, \ldots, f_2, f_1)$ 仅含有 λ 的次数为 $n_1n_2\cdots n_s$ 的项.

作为"互素"概念的一个应用, 我们考虑与代数子流形的一个大范围性质"横截性"有关的问题. 令

$$\begin{cases} F_1(x_1, x_2, x_3) = 0, \\ F_2(x_1, x_2, x_3) = 0 \end{cases}$$

表示一条代数曲线, 并令

$$F_3(x_1, x_2, x_3) = 0$$

表示一个代数曲面. 我们想要知道两者是否横截. 由于曲线的切向量是 $\nabla F_1 \times \nabla F_2$, 而曲面的法向量是 ∇F_3, 故横截性条件为

$$(\nabla F_1 \times \nabla F_2) \cdot \nabla F_3 \neq 0.$$

设 $\{F_1, F_2, F_3\}$ 已转化为一个正常升列 $\{f_1, f_2, f_3\}$, 又令

$$g = (\nabla F_1 \times \nabla F_2) \cdot \nabla F_3,$$

则该曲线与曲面横截的充分必要条件是

$$\mathrm{res}(g, f_3, f_2, f_1) \neq 0,$$

即 g 与 $\{f_1, f_2, f_3\}$ 互素.

否则, 正常情况下切点的数目等于

$$\mathrm{ldegree}(\mathrm{res}(g + \lambda, f_3, f_2, f_1), \lambda),$$

这里 ldegree 含义为 "最低次数".

§ 20 应用相关性判准的几个实例

在几何命题的陈述中常有这样的情况: 满足某个作图语句的图形不是唯一的, 这相当于表示命题假设的方程组 (不妨设它已经化成了一个正常升列) TS 具有多组解. 当 TS 可约时, 它的多组解中可能有的解能够使该命题的结论成立, 而另外一些解却不行. 这时候我们可用 §19 的相关性判准来判定究竟有多少组解能够使结论成立. 这里我们限于讨论等式型的命题, 其结论可以用方程 $g = 0$ 表示.

首先考察一个熟知的命题, 即所谓费尔巴哈 (K.W.Feuerbach) 定理: 与一个三角形的三边相切之圆, 必与此三角形之九点圆相切.

与一个三角形的三边都相切的圆一般有四个, 即一个内切圆和三个旁切圆. 需要知道四者中有几个与三角形的九点圆相切.

对这个命题 (采用适当坐标系代数化之后) 用相关性判准, 其计算机程序 (MAPLE) 如下：

```
fb:=proc()
f1:=-u1^3+4*u1^2*x1+4*u2^2*x1^2*u1-4*u2*x1^2*u1-4*u1
    *x1^2-8*u2^2*x1^3+8*u2*x1^3+4*x1^2*u1^3-8*x1^3*u1^2
    +4*u1*x1^4;
f2:=-2*u2*x1-2*x1*x2+2*x1+2*u1*x2-u1;
f3:=4*u1*x3-u1^2-u2+u2^2;
f4:=4*x4-1-2*u2;
f5:=u1^4-2*u1^2*u2+2*u1^2*u2^2+u2^2-2*u2^3+u2^4+u1^2-
   16*u1^2*x5;
g:=normal(((x1-x3)^2+(x2-x4)^2)^2+x5^2+x1^4
   -2*((x1-x3)^2+(x2-x4)^2)*(x5+x1^2)-2*x5*x1^2);
r4:=resultant(g+T,f5,x5);
r3:=resultant(r4,f4,x4);
r2:=resultant(r3,f3,x3);
r1:=resultant(r2,f2,x2);
r0:=resultant(r1,f1,x1);
lprint('The conjecture is true for', ldegree(r0,T) ,'of
   the', degree(r0,T),'branches')
end;
```

屏幕显示的运行结果是：The conjecture is true for 4 of the 4 branches, 也就是说内切圆和旁切圆四者都与九点圆相切, 即此命题普遍真. 此程序在 586/75 微机上运行时间不到 1 秒.

另一个例子也是经典的：依次以三角形 ABC 的三边为一边作等边三角形 BCA_1 、CAB_1 、ABC_1, 求证 A_1A 、B_1B 、C_1C 三线共点.

由于每个等边三角形都有顺时针和反时针两种作法, 一共得到 8 个不同的图形, 即 8 种不同的情况. 对这个命题 (采用适当坐标系代数化之后) 用相关性判准, 其计算机程序 (MAPLE) 如下：

```
et:=proc()
F1:=2*x1-1;
```

```
F2:=4*x2^2-3;
F3:=x3^2+x4^2-u1^2-u2^2;
F4:=x3^2+x4^2-(x3-u1)^2-(x4-u2)^2;
F5:=(x5-1)^2+x6^2-(u1-1)^2-u2^2;
F6:=(x5-1)^2+x6^2-(x5-u1)^2-(x6-u2)^2;
f1:=F1;
f2:=F2;
f3:=resultant(F3,F4,x4);
f4:=normal(F4);
f5:=resultant(F5,F6,x6);
f6:=normal(F6);
A:=array(1..3,1..3,[[x2-u2,u1-x1,(x2-u2)*u1-(x1-u1)*u2],
   [x4,1-x3,x4],[x6,-x5,0]]);
with(linalg,det):
g:=normal(det(A));
p5:=prem(g,f6,x6);
p4:=prem(p5,f5,x5);
p3:=prem(p4,f4,x4);
p2:=prem(p3,f3,x3);
p1:=prem(p2,f2,x2);
p0:=prem(p1,f1,x1);
if p0=0 then print(`The conjecture is true in general`)
else r5:=resultant(p0+T,f6,x6);
r4:=resultant(r5,f5,x5);
r3:=resultant(r4,f4,x4);
r2:=resultant(r3,f3,x3);
r1:=resultant(r2,f2,x2);
r0:=resultant(r1,f1,x1);
lprint(`The conjecture is true for`, ldegree(r0,T),`of
   the`, degree(r0,T),`branches`);
fi
end;
```

屏幕显示的运行结果是：The conjecture is true for 2 of the 8 branches, 即此命题仅对 8 种情况中之 2 种成立. 此程序在 586/75 微机上运行时间仅 1 秒.

一个颇有难度的命题是所谓泰保－泰勒 (Thébault–Taylor) 定

理. 这是法国几何学家泰保于 1938 年提出的一个猜测, 直到 1983 年才被泰勒所证实. 其第一个证明长达 26 页! 在欧氏几何里这样的难题实属罕见. 这个绝非"等闲"的命题是: 设三角形 ABC 的内心为 w, 外接圆为 Γ. 又设 D 为线段 BC 上的一点; 与 DB, DA 及 Γ 相切之圆的圆心为 w_1; 与 DC, DA 及 Γ 相切之圆的圆心为 w_2. 求证 w 、w_1 、w_2 三点共线.

这里, 命题假设中的作图语句决定的图形究竟有几个, 并非一目了然, 暂且不去管它. 对此命题 (采用适当坐标系代数化之后) 用相关性判准, 其计算机程序 (MAPLE) 如下:

```
tt:=proc()
f1:=4*x1^2*u1^2*u2^4+6*u1^2*u2^4-2*u2^2-2*u2^6
    -2*u2^2*u1^4-2*u2^6*u1^4+u1^2+u1^2*u2^8+4*u2^4*u3
    -4*u2^4*u1^4*u3-4*u2^4*u3^2*u1^2;
f2:=(4*u2^4-8*u2^4*u3*u1^2-4*u1^4*u2^4+8*x1*u1^2*u2^4)
    *x2^2+(-4*u2^2*u1^4+8*u2^4+4*u1^2-4*u2^6*u1^4-4*u2^6
    +8*u1^4*u2^4+4*u1^2*u2^8+8*u1^2*u2^4-8*u1^2*u2^2
    -4*u2^2-8*u2^6*u1^2)*x2+(-2*u1^2+4*u1^2*u2^4
    -2*u1^2*u2^8)*x1-1-8*u2^4*u3+8*u2^2*u3*u1^2+u1^4
    +2*u2^4+4*u2^6*u1^4*u3+4*u1^4*u3*u2^2-12*u2^4*u3
    *u1^2-2*u1^4*u2^4-2*u2^8*u3*u1^2-8*u2^4*u1^4*u3-
    2*u3*u1^2+u1^4*u2^8+4*u3*u2^2-u2^8+8*u2^6*u3*u1^2
    +4*u2^6*u3;
f3:=(4*u2^4-4*u1^4*u2^4-8*u2^4*u3*u1^2-8*x1*u1^2*u2^4)
    *x3^2+(8*u1^4*u2^4+4*u1^2-4*u2^2*u1^4-8*u1^2*u2^2
    -4*u2^6*u1^4+8*u1^2*u2^4-8*u2^6*u1^2+4*u1^2*u2^8
    -4*u2^2+8*u2^4-4*u2^6)*x3+(2*u1^2-4*u1^2*u2^4
    +2*u1^2*u2^8)*x1-1+8*u2^2*u3*u1^2+8*u2^6*u3*u1^2
    +u1^4+2*u2^4-u2^8+4*u3*u2^2-12*u2^4*u3*u1^2-2*u1^4
    *u2^4-8*u2^4*u3-8*u2^4*u1^4*u3-2*u3*u1^2+u1^4*u2^8
    +4*u1^4*u3*u2^2+4*u2^6*u3+4*u2^6*u1^4*u3
    -2*u2^8*u3*u1^2;
f4:=x4*u1*u2+1-u1^2*u2^2;
f5:=x5*u1*u2-u1^2+u2^2;
f6:=(2*u1^2-2*u1^2*u2^4+2*x4*u1^2*u2^2+2*x5*u1^2*u2^2)
    *x6+(-u1^2-u1^2*u2^4)*x5+(-u1^2-u1^2*u2^4)*x4-u1^4
```

```
    +u2^4-1+u1^4*u2^4;
f7:=(-u2^2+u1^2*u2^4-u2^2*u1^4+u1^2)*x7+(u2^2-u2^2*u1^4
    -u1^2+u1^2*u2^4-2*x5*u1^2*u2^2)*x6+(u1^2*u2^4+u1^2)
    *x5-u2^4+u1^4;
g:=(4*x2*u2^4+4*x3*u2^4)*x7+(-4*u2^2-4*u2^6+8*u2^4)*x6
   -4*u2^2-4*u2^6+1+6*u2^4-4*x2*x3*u2^4+u2^8;
r6:=resultant(g+T,f7,x7);
r5:=resultant(r6,f6,x6);
r4:=resultant(r5,f5,x5);
r3:=resultant(r4,f4,x4);
r2:=resultant(r3,f3,x3);
r1:=resultant(r2,f2,x2);
r0:=resultant(r1,f1,x1);
lprint('The conjecture is ture for', ldegree(r0,T),'of
   the',degree(r0,T),'branches')
end;
```

屏幕显示的运行结果是：The conjecture is true for 2 of the 8 branches, 即此命题仅对 8 种情况中之 2 种成立. 此程序在 586/75 微机上运行时间 57 秒. 顺便我们也知道了此命题的假设所决定的图形一般有 8 个.

§ 21 相对单纯分解

考虑方程组

$$\begin{cases} g_1 = x(x-1) = 0, \\ g_2 = y - (x^2+1) = 0, \\ g_3 = x^2 + 5xy + 3y^2 - 6x + 2y - 5 = 0. \end{cases}$$

很明显,

$$\mathrm{prem}(g_3, g_2, g_1) = 16x, \quad \mathrm{res}(g_3, g_2, g_1) = 0.$$

所以 g_3 与 $\{g_1, g_2\}$ 既不是整相关的, 也不是互素的; 这个方程组可

分解为如下两个方程组

$$\begin{cases} x = 0, \\ g_2 = 0, \\ g_3 = 0. \end{cases} \quad 或 \quad \begin{cases} x - 1 = 0, \\ g_2 = 0, \\ g_3 = 0. \end{cases}$$

易知

$$\mathrm{prem}(g_3, g_2, x) = 0, \qquad \mathrm{res}(g_3, g_2, x - 1) = 16.$$

所以 g_3 与 $\{x, g_2\}$ 是整相关的, g_3 与 $\{x - 1, g_2\}$ 是互素的, 因而原方程组的解可表示为 $\{x = 0, y - (x^2 + 1) = 0\}$. 这个简单的实例为本节的概念提供了直观背景.

给定一个三角型方程组:

$$TS: \begin{cases} f_1(x_1) = 0, \\ f_2(x_1, x_2) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_s(x_1, x_2, \ldots, x_s) = 0 \end{cases}$$

和一个多项式方程 $g = g(x_1, x_2, \ldots, x_s) = 0$, 设 TS 对应的三角列是正常升列

$$AC: \quad \{f_1, f_2, \ldots, f_s\},$$

如果方程 $g = 0$ 与方程组 TS 或者是互素的或者是整相关的, 那么就说方程组 TS 相对于方程 $g = 0$ 是 **单纯** 的; 也说正常升列 AC 相对于多项式 g 是 **单纯** 的.

如果正常升列 AC 相对于多项式组 PS 的每个多项式都是单纯的, 我们就说正常升列 AC 相对于多项式组 PS 是 **单纯** 的; 如果 AC 与 PS 中的每一个多项式都是整相关的, 就说 AC 与 PS 是 **整相关** 的; 如果 AC 相对于 PS 是单纯的但不是整相关的, 就说 AC 与 PS 是 **互素** 的.

如上所述, 一般地, 一个三角型方程组与一个多项式方程之间的关系并不简单地是互素或整相关. 为了研究方程组

$$TS: \quad \{f_1 = 0, f_2 = 0, \ldots, f_s = 0, g = 0\},$$

我们要把三角型方程组 $TS:\{f_1=0,f_2=0,\ldots,f_s=0\}$ 分解为多个三角型方程组

$$TS_i:\quad \{f_{i1}=0,f_{i2}=0,\ldots,f_{is}=0\},$$

使得其中的每一个相对于方程 $g=0$ 都是单纯的, 并且分解后这些方程组的解恰好就是原方程组的解, 即

$$\mathrm{Zero}(TS)=\bigcup\mathrm{Zero}(TS_i).$$

这种分解称之为方程组 TS 关于方程 $g=0$ 的 **相对单纯分解**; 也说成是正常升列 AC 关于多项式 g 的 **相对单纯分解**. 这样的分解一般不是唯一的.

确实有一些算法能够实现正常升列关于一个多项式 (或一个多项式组) 的相对单纯分解. 下一节将给出这样的一个算法, 它是基于下面这个定理的.

设

$$AC:\quad\left\{\begin{array}{l}f_1(x_1),\\ f_2(x_1,x_2),\\ \cdots\cdots\cdots\cdots\\ f_s(x_1,x_2,\ldots,x_j)\end{array}\right.$$

是正常升列, 其中 $f_i\in K[x_1,\ldots,x_i]$, $i=1,2,\ldots,j$.

$$f(y)=a_0y^n+a_1y^{n-1}+\cdots+a_{n-1}y+a_n,$$

$$g(y)=b_0y^m+b_1y^{m-1}+\cdots+b_{m-1}y-b_m.$$

是 $K[x_1,x_2,\ldots,x_s]$ 上的两个多项式. 沿用 §10 的记法, $s_i(f,g,y)$ 表示 $f(y)$ 和 $g(y)$ 的第 i 个主子结式:

$$P_0(f,g,y),\quad P_1(f,g,y),\quad \ldots,\quad P_n(f,g,y)$$

(无妨令 $n<m$) 是 $f(x)$ 和 $g(x)$ 的子结式多项式序列 (其定义见 §10). 又设

$$R_i=\mathrm{prem}(s_i(f,g,y),f_j,\ldots,f_1).$$

定理 1 如果

$$\operatorname{res}(a_0, f_j, \ldots, f_1), \quad \operatorname{res}(b_0, f_j, \ldots, f_1)$$

不都等于 0, 且

$$R_0 = \cdots = R_{k-1} = 0, \quad \operatorname{res}(R_k, f_j, \ldots, f_1) \neq 0,$$

则 $P_n(f, g, y)$ 是多项式 $f(y)$ 和 $g(y)$ 在环 $K[x_1, \ldots, x_j]/(f_1, \ldots, f_j)$ 上的最大公因子.

下面给出这个定理的一个证明.

设 $(f_1^*, \ldots, f_j^*)$ 是 $(f_1, \ldots, f_j)$ 的任一不可约分支.

对于 $i = 1, 2, \ldots, j$, 等式 $R_i = 0$ 蕴含

$$s_i(f, g, y) = 0 \quad \operatorname{mod}(f_1, \ldots, f_j),$$

因而

$$s_i(f, g, y) = 0 \quad \operatorname{mod}(f_1^*, \ldots, f_j^*).$$

另一方面, $\operatorname{res}(R_k, f_j, \ldots, f_1) \neq 0$ 意味着

$$s_k(f, g, y) \neq 0 \quad \operatorname{mod}(f_1^*, \ldots, f_j^*).$$

由假设, $\operatorname{res}(a_0, f_j, \ldots, f_1)$ 和 $\operatorname{res}(b_0, f_j, \ldots, f_1)$ 不都等于 0, 所以 a_0 和 b_0 在域 $K[x_1, \ldots, x_j]/(f_1^*, \ldots, f_j^*)$ 中不都等于 0, 因此 $P_k(f, g, y)$ 是多项式 $f(y)$ 和 $g(y)$ 在环

$$K[x_1, \ldots, x_j]/(f_1^*, \ldots, f_j^*)$$

上的最大公因子. 由 $(f_1^*, \ldots, f_j^*)$ 选择的任意性, 即知 $P_k(f, g, y)$ 是多项式 $f(y)$ 和 $g(y)$ 在环 $K[x_1, \ldots, x_j]/(f_1, \ldots, f_j)$ 上的最大公因子. 证毕.

§ 22 相对分解算法

本节给出的算法, 由于使用了吴除法 (即逐次伪除法) 和子结式计算, 有时也称为 **WR分解算法**[42],[43],[50], 其思想源于作者更早的工作[54],[55]. 用它可以作出正常升列 $AC:\{f_1,\ldots,f_s\}$ 相对于多项式 g 的一种相对单纯分解. 这个算法可描述如下.

设若 AC 由一个多项式 f_1 组成, 即 $AC=\{f_1\}$, 则若 $\mathrm{prem}(g,f_1)=0$ 或者 $\mathrm{res}(g,f_1)\neq 0$, 那么 AC 关于 g 是单纯的, 此时没什么要做的; 否则 $\{f_1\}$ 可以分解为 $\{f_{11}\}$ 和 $\{f_{12}\}$, 其中 f_{11} 是 f_1 和 g 的最大公因子, f_{12} 是 f_{11} 除 f_1 的伪商. 一般地, 我们有

步骤 1 计算 g 关于 AC 的余式, 即 $\mathrm{prem}(g,f_s,\ldots,f_1)$. 如果它等于 0, 结束并输出 AC. 否则, 令 $g^*=g$ 或 $g^*=\mathrm{prem}(g,f_s,\ldots,f_1)$. 如果 $\mathrm{res}(g^*,f_s,\ldots,f_1)\neq 0$, 结束并输出 AC.

步骤 2 如果 $\mathrm{res}(g^*,f_s,\ldots,f_1)=0$. 设 k 是使得下式成立的最小整数

$$\mathrm{prem}(s_k(g^*,f_s,x_s),f_{s-1},\ldots,f_1)\neq 0.$$

如果

$$\mathrm{res}(s_k(g^*,f_s,x_s),f_{s-1},\ldots,f_1)\neq 0,$$

则根据 §21 的定理, 在环 $K[x_1,\ldots,x_{s-1}]/(f_1,\ldots,f_{s-1})$ 中可求得 f_s 和 g^* 的最大公因子, 记之为 f_{s1}, 令 f_{s2} 是 f_{s1} 除 f_s 的伪商, 把两个正常升列

$$\{f_1,\ldots,f_{s-1},f_{s1}\};\quad \{f_1,\ldots,f_{s-1},f_{s2}\}$$

分别返回步骤 1 继续分解.

步骤 3 令 k 是使得

$$\mathrm{prem}(s_k(g^*,f_s,x_s),f_{s-1},\ldots,f_1)\neq 0$$

成立的最小整数. 如果

$$\mathrm{res}(s_k(g^*,f_s,x_s),f_{s-1},\ldots,f_1)=0,$$

则以 $\{f_1,\ldots,f_{s-1}\}$ 和 $s_k(g^*,f_s,x_s)$ 分别代替 $\{f_1,\ldots,f_{s-1},f_s\}$ 和 g，返回步骤 1 即作 $\{f_1,\ldots,f_{s-1}\}$ 关于 $s_k(g^*,f_s,x_s)$ 的相对单纯分解；假定这一步已完成，那么把得到的每一个分支，比如说 $\{\bar{f}_1,\ldots,\bar{f}_{s-1}\}$，连同 f_s 一起组成 $\{\bar{f}_1,\ldots,\bar{f}_{s-1},f_s\}$，然后返回步骤 1，作关于 g 的相对单纯分解.

由于上述各步的每次返回，或者使升列中的多项式的次数降低，或者使升列本身的多项式个数减少，所以有限步后必然终止，最后就得到升列 AC 相对于多项式 g 的一种相对单纯分解.

显然我们有

定理 1 每个三角列都可分解为有限个正常升列，使得该三角列所对应三角型方程组的解集等于这些正常升列所对应方程组的解集之并.

[例] 设

$$\begin{aligned}
&f_1 = 4x_1^2 - 3,\\
&f_2 = 2x_2 - 1,\\
&f_3 = x_3 - 1,\\
&f_4 = x_4^2 - 3,\\
&f_5 = 4x_5^2 - 8x_5 + 1,\\
&f_6 = 2x_6 - 4x_5 + 3,\\
&f_7 = ((4 - 2x_1)x_4 + 2x_1 - 3)x_7 + 2 - 2x_1,\\
&f_8 = 2(2 - x_1)x_8 + x_7 - 2,\\
&g = x_5x_8 - x_6x_7.
\end{aligned}$$

求正常升列 $AC:\{f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8\}$ 相对于多项式 g 的单纯分解.

首先

$$\begin{aligned}
g^* &= \mathrm{prem}(g,f_8,\ldots,f_1)\\
&= (36 - 44x_1 + (16 - 8x_1)x_4)x_5 + 36x_1 - 33 \neq 0,
\end{aligned}$$

而 $\mathrm{res}(g^*,f_8,\ldots,f_1) = 0$. 使用三次 Step 3 把 f_8、f_7、f_6 放到一边，

求 $\{f_1, f_2, f_3, f_4, f_5\}$ 关于 g^* 的相对单纯分解. 计算主子结式

$$
\begin{aligned}
s_0(g^*, f_5, x_5) = 4(&16x_1^2x_4^2 - 400x_1^2x_4 - 1388x_1^2 - 64x_1x_4^2 \\
&+1184x_1x_4 + 2328x_1 + 64x_4^2 - 768x_4 - 963).
\end{aligned}
$$

由于

$$
\begin{aligned}
g_1^* &= \mathrm{prem}(s_0(g^*, f_5, x_5), f_4, f_3, f_2, f_1) \\
&= 16((296x_1 - 267)x_4 + 534x_1 - 444) \neq 0,
\end{aligned}
$$

而 $\mathrm{res}(g_1^*, f_4, f_3, f_2, f_1) = 0$, 再次使用 Step 3 把 f_5 放到一边, 求 $\{f_1, f_2, f_3, f_4\}$ 关于 g_1^* 的相对单纯分解. 计算主子结式

$$s_0(g_1^*, f_4, x_4) = 89232(4x_1^2 - 3).$$

因为 $\mathrm{prem}(s_0(g_1^*, f_4, x_4), f_3, f_2, f_1) = 0$, 考虑下一个主子结式

$$s_1(g_1^*, f_4, x_4) = 16(296x_1 - 267).$$

$\mathrm{res}(s_1(g_1^*, f_4, x_4), f_3, f_2, f_1) \neq 0$. 根据 Step 2, f_4 和 g_1^* 有最大公因子, 即 g_1^* 自身, 令

$$f_{41} = (296x_1 - 267)x_4 + 534x_1 - 444$$

且 f_{42} 为 f_{41} 除 f_4 的伪商, 即

$$f_{42} = (296x_1 - 267)x_4 - 534x_1 + 444.$$

接下来分别求 $\{f_1, f_2, f_3, f_{41}, f_5\}$ 和 $\{f_1, f_2, f_3, f_{42}, f_5\}$ 相对于 g^* 的单纯分解. 因为 $\mathrm{prem}(s_0(g^*, f_5, x_5), f_{41}, f_3, f_2, f_1) = 0$, 考虑下一个主子结式

$$s_1(g^*, f_5, x_5) = 36 - 44x_1 + (16 - 8x_1)x_4.$$

$\mathrm{res}(s_1(g^*, f_5, x_5), f_{41}, f_3, f_2, f_1) \neq 0$. 根据 Step 2, f_5 和 g^* 有最大公因子, 即 g^* 自身. 令 $f_{51} = g^*$, 且 f_{52} 为 f_{51} 除 f_5 的伪商, 即

$$f_{52} = (36 - 44x_1 + (16 - 8x_1)x_4)x_5 - (32 - 16x_1)x_4 + 52x_1 - 39.$$

这样，我们就把 $AC:\{f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8\}$ 分解为如下三个正常升列：

$$\begin{array}{ll} AC_1: & \{f_1,f_2,f_3,f_{41},f_{51},f_6,f_7,f_8\}, \\ AC_2: & \{f_1,f_2,f_3,f_{41},f_{52},f_6,f_7,f_8\}, \\ AC_3: & \{f_1,f_2,f_3,f_{42},f_5,f_6,f_7,f_8\}. \end{array}$$

用 Step 1 检验知，g 与 AC_1 整相关，g 与 AC_2 和 AC_3 互素.

我们编有 (用 MAPLE 写的) 该算法的通用程序 "**WR**"，可对给定的正常升列相对于一个给定的多项式 g 作单纯分解，其输出是关于 g 单纯的若干升列 (见本书附录). 该程序在 PC 机和工作站上均可运行，机器内存最好在 8 兆以上.

§ 23 相对单纯分解的一个实例

为了说明这一算法的效率，我们将举一个足够复杂的例子. 前面我们已经提到过泰保 - 泰勒定理，并知该命题对题设 8 种情况中的 2 种成立. 仅仅由相关性判准无法区分究竟在哪些情况下命题为真，故须将 (表示假设的) 升列 $\{f_1,\cdots,f_7\}$ 相对于 (表示结论的) 多项式 g 作单纯分解.

当然，也可采取另一种做法，即将升列 $\{f_1,\cdots,f_7\}$ 作不可约分解. 在 [8] 中正是这样给出了泰保 - 泰勒定理的第一个计算机证明. 其程序在一 SYMBOLICS 3600 机上的运行时间是 44 小时. 考虑到这一猜测在历史上曾 44 年悬而未决 (且泰勒所给的证明长达 26 页)，则 44 机时也不算多. 在 [35] 中采用更多的几何处理改进之后，将运行时间缩短到 6 小时.

以下是我们对这一问题的处理及算法运用情况.

由于相对单纯分解无须调用任何意义下的因式分解算法，加之我们对此几何命题采取了简单得多的代数表示，在 PC 586/75 机上很快就将升列 $\{f_1,\cdots,f_7\}$ 分解为关于 g 单纯的 4 个分支. 下面这个交互式 (MAPLE) 程序是全过程的运行记录. 累计用的 CPU 时间仅 10 秒！

```
f1:=4*x1^2*u1^2*u2^4+6*u1^2*u2^4-2*u2^2-2*u2^6
    -2*u2^2*u1^4-2*u2^6*u1^4+u1^2+u1^2*u2^8+4*u2^4*u3
    -4*u2^4*u1^4*u3-4*u2^4*u3^2*u1^2;
f2:=(4*u2^4-8*u2^4*u3*u1^2-4*u1^4*u2^4+8*x1*u1^2*u2^4)
    *x2^2+(-4*u2^2*u1^4+8*u2^4+4*u1^2-4*u2^6*u1^4-4*u2^6
    +8*u1^4*u2^4+4*u1^2*u2^8+8*u1^2*u2^4-8*u1^2*u2^2
    -4*u2^2-8*u2^6*u1^2)*x2+(-2*u1^2+4*u1^2*u2^4
    -2*u1^2*u2^8)*x1-1-8*u2^4*u3+8*u2^2*u3*u1^2+u1^4
    +2*u2^4+4*u2^6*u1^4*u3+4*u1^4*u3*u2^2
    -12*u2^4*u3*u1^2-2*u1^4*u2^4-2*u2^8*u3*u1^2
    -8*u2^4*u1^4*u3-2*u3*u1^2+u1^4*u2^8+4*u3*u2^2-u2^8
    +8*u2^6*u3*u1^2+4*u2^6*u3;
f3:=(4*u2^4-4*u1^4*u2^4-8*u2^4*u3*u1^2-8*x1*u1^2*u2^4)
    *x3^2+(8*u1^4*u2^4+4*u1^2-4*u2^2*u1^4-8*u1^2*u2^2
    -4*u2^6*u1^4+8*u1^2*u2^4-8*u2^6*u1^2+4*u1^2*u2^8
    -4*u2^2+8*u2^4-4*u2^6)*x3+(2*u1^2-4*u1^2*u2^4
    +2*u1^2*u2^8)*x1-1+8*u2^2*u3*u1^2+8*u2^6*u3*u1^2
    +u1^4+2*u2^4-u2^8+4*u3*u2^2-12*u2^4*u3*u1^2
    -2*u1^4*u2^4-8*u2^4*u3-8*u2^4*u1^4*u3-2*u3*u1^2
    +u1^4*u2^8+4*u1^4*u3*u2^2+4*u2^6*u3+4*u2^6*u1^4*u3-
    2*u2^8*u3*u1^2;
f4:=x4*u1*u2+1-u1^2*u2^2;
f5:=x5*u1*u2-u1^2+u2^2;
f6:=(2*u1^2-2*u1^2*u2^4+2*x4*u1^2*u2^2+2*x5*u1^2*u2^2)
    *x6+(-u1^2-u1^2*u2^4)*x5+(-u1^2-u1^2*u2^4)*x4-u1^4
    +u2^4-1+u1^4*u2^4;
f7:=(-u2^2+u1^2*u2^4-u2^2*u1^4+u1^2)*x7+(u2^2-u2^2*u1^4
    -u1^2+u1^2*u2^4-2*x5*u1^2*u2^2)*x6+(u1^2*u2^4+u1^2)
    *x5-u2^4+u1^4;
g:=(4*x2*u2^4+4*x3*u2^4)*x7+(-4*u2^2-4*u2^6+8*u2^4)*x6
    -4*u2^2-4*u2^6+1+6*u2^4-4*x2*x3*u2^4+u2^8;
i1:=sprem(sprem(sprem(sprem(sprem(sprem(sprem(g,f7,x7),
    f6,x6),f5,x5),f4,x4),f3,x3),f2,x2),f1,x1):
i2:=resultant(i1,f3,x3):
i3:=primpart(i2,{x2,x1}):
i4:=sprem(sprem(i3,f2,x2),f1,x1):
i5:=resultant(i3,f2,x2):
i6:=primpart(i5,x1):
```

```
i7:=resultant(i6,f1,x1):
i8:=primpart(i4,{x2,x1}):
i9:=resultant(i8,f2,x2):
i10:=primpart(i9,x1):
i11:=sprem(i10,f1,x1):
i12:=sprem(coeff(i8,x2,1),f1,x1):
i13:=resultant(coeff(i8,x2,1),f1,x1):
i14:=sprem(f2,i8,x2,'m1','q1'):
i15:=sprem(sprem(sprem(sprem(sprem(sprem(sprem(g,f7,x7),
     f6,x6),f5,x5),f4,x4),f3,x3),i8,x2),f1,x1):
i16:=resultant(g,f7,x7):
i17:=resultant(i16,f6,x6):
i18:=primpart(i17,{x5,x4,x3,x2,x1}):
i19:=resultant(i18,f5,x5):
i20:=primpart(i19,{x4,x3,x2,x1}):
i21:=resultant(i20,f4,x4):
i22:=primpart(i21,{x3,x2,x1}):
i23:=resultant(i22,f3,x3):
i24:=primpart(i23,{x2,x1}):
i25:=resultant(i24,i8,x2):
i26:=primpart(i25,x1):
i27:=resultant(i26,f1,x1):
i28:=primpart(i15,{x3,x2,x1}):
i29:=resultant(i28,f3,x3):
i30:=sprem(sprem(i29,i8,x2),f1,x1):
i31:=sprem(sprem(coeff(i28,x3,1),i16,x2),f1,x1):
i32:=resultant(resultant(coeff(i28,x3,1),i8,x2),f1,x1):
i33:=sprem(f3,i28,x3,'m2','q2'):
f21:=i8: f22:=primpart(q1,x2): f31:=i28: f32:=q2:
```

这样, 我们就得到原升列的 4 个单纯分支:

$$
\begin{aligned}
C_1: &\quad \{f_1, f_{21}, f_{31}, f_4, f_5, f_6, f_7\},\\
C_2: &\quad \{f_1, f_{21}, f_{32}, f_4, f_5, f_6, f_7\},\\
C_3: &\quad \{f_1, f_{22}, f_{31}, f_4, f_5, f_6, f_7\},\\
C_4: &\quad \{f_1, f_{22}, f_{32}, f_4, f_5, f_6, f_7\}.
\end{aligned}
$$

其中

```
f21:=u1^2+u1*u2-u1^3*u2-u2^2-2*u1^2*u2^2-u1^4*u2^2
     +u1*u2^3-u1^3*u2^3+2*u2^4+2*u1^2*u2^4+2*u1^4*u2^4
     -u1*u2^5+u1^3*u2^5-u2^6-2*u1^2*u2^6-u1^4*u2^6
     -u1*u2^7+u1^3*u2^7+u1^2*u2^8-2*u1*u2^3*u3
     -2*u1^3*u2^3*u3+2*u1*u2^5*u3+2*u1^3*u2^5*u3
     +2*u1*u2^3*x1+2*u1^3*u2^3*x1-2*u1*u2^5*x1-
     2*u1^3*u2^5*x1+2*u2^4*x2-2*u1^4*u2^4*x2
     -4*u1^2*u2^4*u3*x2+4*u1^2*u2^4*x1*x2;
f22:=-u1^2+u1*u2-u1^3*u2+u2^2+2*u1^2*u2^2+u1^4*u2^2
     +u1*u2^3-u1^3*u2^3-2*u2^4-2*u1^2*u2^4-2*u1^4*u2^4
     -u1*u2^5+u1^3*u2^5+u2^6+2*u1^2*u2^6+u1^4*u2^6
     -u1*u2^7+u1^3*u2^7-u1^2*u2^8-2*u1*u2^3*u3
     -2*u1^3*u2^3*u3+2*u1*u2^5*u3+2*u1^3*u2^5*u3
     +2*u1*u2^3*x1+2*u1^3*u2^3*x1-2*u1*u2^5*x1-
     2*u1^3*u2^5*x1-2*u2^4*x2+2*u1^4*u2^4*x2
     +4*u1^2*u2^4*u3*x2-4*u1^2*u2^4*x1*x2;
f31:=1+u1^2-2*u2^2-2*u1^2*u2^2+u2^4+u1^2*u2^4-2*u1*u2*u3
     +2*u1*u2^3*u3+2*u1*u2*x1-2*u1*u2^3*x1+2*u1*u2*x3
     -2*u2^2*x3+2*u1^2*u2^2*x3-2*u1*u2^3*x3;
f32:=2*u1^3-2*u1^2*u2+2*u1^4*u2-2*u1*u2^2-6*u1^3*u2^2-
     2*u1^5*u2^2+u2^3+u1^2*u2^3-u1^4*u2^3-u1^6*u2^3
     +6*u1*u2^4+8*u1^3*u2^4+6*u1^5*u2^4-2*u2^5
     +2*u1^2*u2^5-2*u1^4*u2^5+2*u1^6*u2^5-6*u1*u2^6
     -8*u1^3*u2^6-6*u1^5*u2^6+u2^7+u1^2*u2^7-u1^4*u2^7
     -u1^6*u2^7+2*u1*u2^8+6*u1^3*u2^8+2*u1^5*u2^8
     -2*u1^2*u2^9+2*u1^4*u2^9-2*u1^3*u2^10
     +2*u1^2*u2^3*u3+2*u1^4*u2^3*u3+2*u1*u2^4*u3
     -2*u1^5*u2^4*u3-4*u1^2*u2^5*u3-4*u1^4*u2^5*u3
     -2*u1*u2^6*u3+2*u1^5*u2^6*u3+2*u1^2*u2^7*u3
     +2*u1^4*u2^7*u3-4*u1^3*u2^4*u3^2+4*u1^3*u2^6*u3^2
     +2*u1^2*u2^3*x1+2*u1^4*u2^3*x1-2*u1*u2^4*x1
     +2*u1^5*u2^4*x1-4*u1^2*u2^5*x1-4*u1^4*u2^5*x1
     +2*u1*u2^6*x1-2*u1^5*u2^6*x1+2*u1^2*u2^7*x1
     +2*u1^4*u2^7*x1+4*u1^3*u2^4*x1^2-4*u1^3*u2^6*x1^2
     +2*u1*u2^4*x3-2*u1^5*u2^4*x3-2*u2^5*x3
     +2*u1^2*u2^5*x3+2*u1^4*u2^5*x3-2*u1^6*u2^5*x3
```

```
-2*u1*u2^6*x3+2*u1^5*u2^6*x3-4*u1^3*u2^4*u3*x3
+4*u1^2*u2^5*u3*x3-4*u1^4*u2^5*u3*x3+4*u1^3*u2^6*u3
*x3-4*u1^3*u2^4*x1*x3+4*u1^2*u2^5*x1*x3-4*u1^4*u2^5
*x1*x3+4*u1^3*u2^6*x1*x3;
```

计算机显示 g 仅与 C_1 整相关, 而与 C_2, C_3, C_4 等三分支互素. 也就是说, 泰保 - 泰勒定理仅在 C_1 表述的前提下成立.

如果不采取人机对话方式, 而是用 §22 中提到的 **WR** 通用程序来解此问题, 在同一台 PC 机上需时仅 8 秒！这多少有些出人意料.

§24 非退化条件

前面有关三角型方程组的论述, 所考虑的基域是 $K(u_1, \ldots, u_r)$ 而不是 K, 我们已在本章开始时 (参见本书第 39 页) 作了约定. 在很多情况下人们只关心一般的参数解, 这时前述理论是完全的. 可是这并不是事情的全部, 也有一些情形, 人们需要了解方程组的所有解, 而无论变元或者参变元.

为了直观地理解这两者之间的差异, 让我们来考察下面这个简单例子：

$$\begin{aligned} f_1 &= x_1 - u, \\ f_2 &= ux_2 - x_1^2, \\ g &= x_2 - u. \end{aligned}$$

显然, $\{f_1, f_2\}$ 是正常升列, 且

$$g = \frac{1}{u}(x_1 + u)f_1 + \frac{1}{u}f_2.$$

可是 g 却不可能既表为 f_1 和 f_2 的组合, 而又不带分式. 换句话说, g 属于 f_1 和 f_2 在 $K(u)[x_1, x_2]$ 中生成的理想, 但不属于 f_1 和 f_2 在 $K[u, x_1, x_2]$ 中生成的理想.

方程组 $\{f_1=0, f_2=0\}$ 的所有解可以表示为

$$\begin{cases} x_1-u=0, \\ x_2-u=0 \end{cases} \quad \text{或} \quad \begin{cases} u=0, \\ x_1-u=0, \\ x_2 \text{ 任意}. \end{cases}$$

如果我们对参变元 u 的特殊取值不感兴趣，则只需保留第一组解；此时方程组 $\{f_1=0, f_2=0, g=0\}$ 与方程组 $\{f_1=0, f_2=0\}$ 同解；反之则不然.

由此可见，当我们要对方程的所有未知元一视同仁而不作参变元和变元这样的区分的时候，确实还有一些事情需要澄清. 本节给出一个预备性的定理，进一步的论述将在下一节给出.

给定一个正常升列 $AC:\{f_1, f_2, \ldots, f_s\}$，不妨令

$$f_i = c_{i0}x_i^{n_i} + c_{i1}x_i^{n_i-1} + \cdots + c_{in_i}, \qquad i=1,2,\cdots,s$$

多项式组 $\{f_1, f_2, \ldots, f_s\}$ 的一个零点 $(\tilde{u}_1, \cdots, \tilde{u}_r, \tilde{x}_1, \cdots, \tilde{x}_s)$，如果使得所有

$$c_{i0} \neq 0, \qquad i=1,2,\cdots,s$$

则称其为一个 **正常零点**，此时我们说它满足 **非退化条件**[34],[8]；如果使得对于每个 $i\,(i=1,2,\cdots,s)$，在

$$c_{i0}, \quad c_{i1}, \quad \cdots, \quad c_{in_i}$$

中至少有一个不为 0，则称其为一个 **拟正常零点**，此时我们说它满足 **弱非退化条件** [49].

显然每个正常零点必是一个拟正常零点，反之则不成立. 我们有

定理 1 给定正常升列 $AC:\{f_1, \ldots, f_s\}$ 和多项式 $g=g(u, x_1, \ldots, x_s)$，如果 AC 的每一个正常零点都是 g 的一个零点，那么 AC 的每一个拟正常零点也都是 g 的一个零点.

为了证明这个定理，需要如下两个引理.

引理 1 给定复数域上的多项式

$$f(z)=a_nz^n+\cdots+a_kz^k+\cdots+a_1z+a_0,$$

如果 $a_n\neq 0$, $a_k\neq 0$, $n\geq k\geq 1$, 则 $f(x)$ 必有一根 z_1 使得

$$|z_1|\leq\sqrt[k]{C_n^k\left|\frac{a_0}{a_k}\right|}.$$

证明 设 z_1, z_2, $\cdots$, z_n 是 $f(x)$ 的所有 n 个根, 且

$$|z_1|\leq|z_2|\leq\cdots\leq|z_n|.$$

由根与系数的关系可得

$$C_n^k|z_{k+1}z_{z+2}\cdots z_n|\geq\left|\frac{a_k}{a_n}\right|$$

及

$$|z_1z_2\cdots z_n|=\left|\frac{a_0}{a_n}\right|.$$

由此易知

$$|z_1z_2\cdots z_k|\leq C_n^k\left|\frac{a_0}{a_n}\right|.$$

因而

$$|z_1|\leq\sqrt[k]{C_n^k\left|\frac{a_0}{a_k}\right|}.$$

引理 2 给定复数域上的多项式

$$g(z)=b_kz^k+\cdots+b_1z+b_0,\qquad\qquad (k\geq 1,\ b_k\neq 0)$$

对于任意给定的整数 $n\geq k$ 及正实数 ε, 必定存在一个正实数 δ, 使得 $g(z)$ 的每个根的 ε- 邻域必定包含多项式

$$f(z)=a_nz^n+\cdots-a_kz^k+\cdots+a_1z+a_0$$

的一个根, 其中

$$\begin{aligned} |a_i| \le \delta, \qquad & i = k+1, \ldots, n. \\ |a_j - b_j| \le \delta, \qquad & j = 0, 1, \ldots, k. \end{aligned}$$

证明 设 z_0 是 $g(z)$ 的一个根. 作平移变换

$$z \mapsto z + z_0.$$

$g(z)$ 和 $f(z)$ 分别被变换成为

$$\tilde{g}(z) = \tilde{b}_k z^k + \tilde{b}_{k-1} z^{k-1} + \cdots + \tilde{b}_1 z$$

和

$$\tilde{f}(x) = \tilde{a}_n z^n + \tilde{a}_{n-1} z^{n-1} + \cdots + \tilde{a}_1 z + f(z_0),$$

其中

$$\tilde{b}_j = \sum_{i=j}^{k} b_i C_i^j z_0^{i-j}, \qquad \tilde{a}_j = \sum_{i=j}^{n} a_i C_i^j z_0^{i-j}.$$

令 $M = |z_0|^n + |z_0|^{n-1} + \cdots + |z_0| + 1$, 则

$$|f(z_0)| \le (|z_0|^n + |z_0|^{n-1} + \cdots + |z_0| + 1)\delta = M\delta. \tag{24.1}$$

又因

$$\begin{aligned} |\tilde{a}_k| &= \left| \sum_{i=k}^{n} a_i C_i^k z_0^{i-k} \right| \\ &\ge |a_k| - \left| \sum_{i=k+1}^{n} a_i C_i^k z_0^{i-k} \right| \\ &\ge |b_k| - \delta - \delta \sum_{i=k+1}^{n} C_i^k |z_0|^{i-k}, \end{aligned}$$

令 $\delta \le |b_k| / (2(1 + \sum_{i=k+1}^{n} C_i^k |z_0|^{i-k}))$, 则

$$|\tilde{a}_k| \ge \frac{1}{2} |b_k|. \tag{24.2}$$

由 (24.1) 式、(24.2) 式及引理 1 可知, 在 $\tilde{f}(z)$ 的所有根中至少有一个, 设为 z_1, 使得

$$|z_1| \leq \sqrt[k]{C_n^k \left|\frac{f(z_0)}{\bar{a}_k}\right|} \leq \sqrt[k]{C_n^k \frac{2M\delta}{|b_k|}}.$$

因此, 给定 $\varepsilon > 0$, 不妨令

$$\delta \leq \min\left\{\frac{|b_k|}{2(1+\sum\limits_{i=k+1}^{n} C_i^k |z_0|^{i-k})}, \frac{\varepsilon^k |b_k|}{2MC_n^k}\right\},$$

那么, 在 $\bar{g}(z)$ 的零根的 ε- 邻域必定包含多项式 $\tilde{f}(z)$ 的一根, 即 z_0 的 ε- 邻域必定包含多项式 $f(z)$ 的一个根.

以下是定理 1 的证明:

如若不然, 即有 AC 的一个拟正常零点 $(u^*, x_1^*, \ldots, x_s^*)$ 使得

$$g(u^*, x_1^*, \ldots, x_s^*) \neq 0.$$

则存在一个 $\varepsilon > 0$, 使得当

$$|u - u^*| < \varepsilon, \quad |x_1 - x_1^*| < \varepsilon, \quad , \ldots, \quad , |x_s - x_s^*| < \varepsilon$$

时

$$g(u, x_1, \ldots, x_s) \neq 0.$$

根据引理 2, 我们可以找到一个 δ_s, 使得对于任意满足

$$|u - u^*| < \delta_s, \quad |x_1 - x_1^*| < \delta_s, \quad , \ldots, \quad , |x_{s-1} - x_{s-1}^*| < \delta_s$$

及

$$c_s(u, x_1, \ldots, x_{s-1}) \neq 0$$

的 $(u, x_1, \ldots, x_{s-1})$, 必存在一个数 x_s, 满足

$$|x_s - x_s^*| < \varepsilon, \qquad f_s(u, x_1, \ldots, x_s) = 0.$$

类似地, 我们可以找到一个 δ_{s-1}, 使得对于任意满足

$$|u-u^*|<\delta_{s-1},\quad |x_1-x_1^*|<\delta_{s-1},\quad \ldots,\quad |x_{s-2}-x_{s-2}^*|<\delta_{s-1}$$

及

$$c_{s-1}(u,x_1,\ldots,x_{s-2})\neq 0$$

的 $(u,x_1,\ldots,x_{s-2})$, 必存在一个数 x_{s-1}, 满足

$$|x_{s-1}-x_{s-1}^*|<\min\{\delta_s,\varepsilon\},\qquad f_{s-1}(u,x_1,\ldots,x_{s-1})=0.$$

如此继续下去, 我们可以一个一个地依次得到数 δ_s, δ_{s-1}, $\ldots$, δ_2, 最后我们可以找到一个 δ_1, 使得只要

$$|u-u^*|<\delta_1,\quad c_1(u_1)\neq 0,$$

那么必存在一个数 x_1, 满足

$$|x_1-x_1^*|<\min\{\delta_2,\delta_3,\ldots,\delta_s,\varepsilon\},\qquad f_1(u,x_1)=0.$$

现在令

$$\delta=\min\{\delta_1,\delta_2,\ldots,\delta_s,\varepsilon\}.$$

因正常升列 AC 的正常参数点在 C^d 中是稠密的, 故必有一个正常参数点 u_0 适合 $|u_0-u_0^*|<\delta$. 由 u_0 开始, 根据上面的讨论, 我们可以顺次找到一列数 x_1^0, x_2^0, $\ldots$, x_s^0, 使得

$$(u_0,x_1^0,\ldots,x_s^0)$$

是 AC 的一个零点, 且对于 $j=1,2,\ldots,s$, $|x_j^0-x_j^*|<\varepsilon$. 它显然是 AC 的一个正常零点 (因据如上的选取, u_0 是正常参数点), 因而由命题假设, 它是 g 的一个零点. 但前面的反证假设断言 $(u^*,x_1^*,\ldots,x_s^*)$ 点的 ε- 邻域没有 g 的零点, 这是一个矛盾. 证毕.

§ 25 解的结构

以下总设基域为 K, 而不是 $K(u_1,\ldots,u_r)$, 也就是说, 当我们提到多项式的时候, 其系数是取自 K 的.

现在我们需要重新考虑一个问题, 即三角型方程组

$$TS:\begin{cases} f_1=f_1(u_1,\ldots,u_r;x_1)=0,\\ f_2=f_2(u_1,\ldots,u_r;x_1,x_2)=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ f_s=f_s(u_1,\ldots,u_r;x_1,x_2,\ldots,x_s)=0\end{cases}$$

与多项式方程 $g=g(u_1,\cdots,u_r,x_1,\cdots,x_s)=0$(或多项式方程组 PS) 有怎样的关系? 或者说, 考虑方程组

$$GS:\begin{cases} f_1=f_1(u_1,\ldots,u_r;x_1)=0,\\ f_2=f_2(u_1,\ldots,u_r;x_1,x_2)=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ f_s=f_s(u_1,\ldots,u_r;x_1,x_2,\ldots,x_s)=0,\\ g=g(u_1,\cdots,u_r,x_1,\cdots,x_s)=0\end{cases}\qquad (g\in PS)$$

的解的结构.

不失一般性, 下面考虑 PS 仅有一个多项式的情形.

代替方程组的解, 我们考虑相应多项式组的零点. 我们要把多项式组 GS 化为一些三角列 $TS_i(i=1,\cdots,k)$, 这些三角列的零点分别除去相应另外一些三角列 TS_i' $(i=1,\cdots,k)$ 的零点, 合起来恰好是 GS 的零点. 即有**零点结构定理**:

定理 1

$$\mathrm{Zero}(GS)=\bigcup_{i=1}^{k}\mathrm{Zero}(TS_i\setminus TS_i').$$

这里 $\mathrm{Zero}(GS)$ 表示多项式组 GS 公共零点之集, 而 $\mathrm{Zero}(TS\setminus TS')$ 表示多项式组 TS 公共零点之集与多项式组 TS' 公共零点之集的差集.

首先让我们考虑 $\{f_1, f_2, \ldots, f_s\}$ 是正常升列的情形. 不妨暂时仍把 $u_1, \ldots, u_r$ 看作参变元, 则用 WR 相对分解算法可以把 $\{f_1, f_2, \ldots, f_s\}$ 关于 g 作相对单纯分解; 我们仍用 $\{f_1, f_2, \ldots, f_s\}$ 来记这样的一个单纯分支. 现在就只要考虑两种极端情形了, 即互素或整相关.

$\{f_1, f_2, \ldots, f_s\}$ 与 g 互素意味着

$$f_0 = \text{res}(g, f_s, \cdots, f_2, f_1). \qquad 0 \neq f_0 \in K[u_1, \cdots, u_r]$$

如果 $f_0 \in K$, 那么多项式组 $\{f_1, f_2, \ldots, f_s, g\}$ 显然没有公共零点. 否则不妨假设 u_r 真的在 f_0 中出现, 问题就化成了考虑三角列

$$TS_0: \quad \begin{cases} f_0 = f_0(u_1, \cdots, u_{r-1}, u_r), \\ f_1 = f_1(u_1, \cdots, u_{r-1}, u_r, x_1), \\ f_2 = f_2(u_1, \cdots, u_{r-1}, u_r, x_1, x_2), \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ f_s = f_s(u_1, \cdots, u_{r-1}, u_r, x_1, x_2, \cdots, x_s) \end{cases} \tag{25.1}$$

与多项式 g 的关系. 我们有

$$\text{Zero}(\{g, f_1, f_2, \cdots, f_s\}) = \text{Zero}(\{f_0, f_1, f_2, \cdots, f_s\}).$$

即

$$\text{Zero}(GS) = \text{Zero}(TS_0).$$

这是一个较为简单的情形, 因为"参变元"减少了一个.

考虑 $\{f_1, f_2, \cdots, f_s\}$ 与 g 整相关的情形. 沿用上一节的记号, 如果方程组

$$c_{10} = 0, \quad c_{11} = 0, \quad \cdots, \quad c_{1n_1} = 0$$

有解且满足方程 $f_0 = f_0(u_1, \cdots, u_{r-1}, u_r) = 0$(比如可以令 $f_0 = c_{10}$), 则根据 §24 的定理 1, 进一步需要考虑的只是形如 (25.1) 的三角列与多项式 g 的关系了; 此时我们有

$$\text{Zero}(GS) = \text{Zero}(TS \setminus TS_0) \bigcup \text{Zero}(\{g\} \cup TS_0).$$

递推地, 不妨假定对于每个 $i(i=1,2,\cdots,s-1)$, 方程组

$$\{f_1=0,\cdots,f_{i-1}=0,c_{i0}=0,c_{i1}=0,\cdots,c_{in_i}=0\}$$

无解. 如果方程组

$$\{f_1=0,\cdots,f_{s-1}=0,c_{s0}=0,c_{s1}=0,\cdots,c_{sn_s}=0\}$$

无解, 则根据 §24 的定理 1 知

$$\mathrm{Zero}(GS)=\mathrm{Zero}(TS).$$

否则, 考虑三角列

$$TS_{-1}:\begin{cases} f_1=f_1(u_1,\cdots,u_r,x_1),\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ f_{s-1}=f_{s-1}(u_1,\cdots,u_r,x_1,\cdots,x_{s-1})\end{cases}$$

与多项式 $h=c_{sj}(j=0,1,\cdots,n_s)$ 的关系. 这是一个较为简单的情形, 因为"变元"减少了一个 (三角列中减少了一个多项式). 假设在这种简单情形时得到

$$\mathrm{Zero}(\{c_{s0},\cdots,c_{sn_s},f_1,\cdots,f_{s-1}\})=\bigcup_{i=1}^{k}\mathrm{Zero}(TS_i\setminus TS_i').$$

其中 TS 及 TS' 都是关于变元 $x_1,\cdots,x_{s-1}$ 的三角列, 则根据 §24 的定理 1, 可知:

$$\mathrm{Zero}(GS)=\bigcup_{i=1}^{k}\mathrm{Zero}(TS_i\cup\{g\}\setminus TS_i').$$

如果 $\{f_1,f_2,\cdots,f_s\}$ 不是正常升列. 设若 k 是使得

$$\mathrm{res}(c_{i0},f_{i-1},\cdots,f_1)\equiv 0$$

成立的最小整数, 我们可以关于 c_{i0} 作正常升列 $\{f_1,\cdots,f_{i-1}\}$ 的相对单纯分解, 每一个分支与 $f_i,\cdots,f_s$ 一起, 形成较为简单的三

角列. 对每个这种简单的三角列重复这个过程. 有限次后, 必然得到有限多个正常升列. 考虑这些正常升列与三角列中每一个多项式以及与多项式 g 的关系, 就又回到了上面所论情形 (参见 §22 定理 1).

这样就完成了定理 1 的证明, 且证明过程是构造性的.

第 4 章

一般多项式方程组

本章讨论一般的非线性代数方程组．正如我们在第 1 章所说的，要从给定的方程组构造出一个所谓导出方程组，然后把这个导出方程组看作是诸未知元各不同幂积的线性方程组，从而利用已得到充分发展的、丰富的线性方法来研究原来的非线性方程组．

求解线性方程组有两种重要的方法，一种是有显式表达的克莱姆法则，一种是依次消去变元而把方程组三角化的所谓高斯消去法．类似地，求解非线性代数方程组也有两个方向，一是把原方程组化为一些三角型方程组，从而把一般的问题归结为现已熟知的三角型方程组的问题，另一方向对应于线性情形的克莱姆法则，希望用显式来表达方程组的解，在此方向虽然也有一些结果，但在通往一般性结果的道路上仍有一些困难有待征服．

用什么样的方法从一般的方程组构造出导出方程组，这对于算法的效率来说是个关键．本章以大量的篇幅介绍了几种结式方法．它们都有各自的优缺点，而且目前仍在发展之中．

与第 2 章一样，我们一般假定所论基域是 K 上 $u_1,\cdots,u_r$ 的有理函数域 $K(u_1,\cdots,u_r)$．有时简单地也把 $K(u_1,\cdots,u_r)$ 写为 K．为了明确起见，我们给出如下定义．

给定代数方程组

$$PS:\begin{cases} p_1(u_1,\cdots,u_r;x_1,\cdots,x_s)=0,\\ p_2(u_1,\cdots,u_r;x_1,\cdots,x_s)=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ p_n(u_1,\cdots,u_r;x_1,\cdots,x_s)=0.\end{cases}$$

如果关于参变元 $u_1, \cdots, u_r$ 的代数函数 $x_i(u_1,\cdots,u_r)$ 满足

$$PS: \begin{cases} p_1(u_1,\cdots,u_r;x_1(u_1,\cdots,u_r),\cdots,x_s(u_1,\cdots,u_r))=0,\\ p_2(u_1,\cdots,u_r;x_1(u_1,\cdots,u_r),\cdots,x_s(u_1,\cdots,u_r))=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ p_n(u_1,\cdots,u_r;x_1(u_1,\cdots,u_r),\cdots,x_s(u_1,\cdots,u_r))=0, \end{cases}$$

那么我们把

$$(x_1(u_1,\cdots,u_r),\cdots,x_s(u_1,\cdots,u_r))$$

称为方程组 PS 的 **符号解** 或 **参数解**; 也称其为多项式组 $p_1, p_2, \cdots, p_n$ 的 **参数零点**. 在本章中如不作特别说明, 一般总是在这个意义下研究方程组的解或多项式组的零点的; 当然, 如所周知, 这并没有失去一般性.

很多时候, 我们要把某些或全部变元取值为 0 的那些零点放到一边, 而只考虑全部变元的取值都非零的那些零点. 这样的零点称之为 **无挠零点** (在方程组的情形则是 **无挠解**).

§ 26 一 个 例 子

让我们还是从一个例子谈起.

给定有理参数曲面

$$(x,y,z)=\left(\frac{s^2t-t-s^2-1}{s^2+s^2t},\frac{s^2t-t+s}{s^2+s^2t},\frac{2s^2-2t-2}{s^2+s^2t}\right),$$

计算其逆映射.

这个参数曲面可用多项式方程组表示为

$$\begin{aligned} p_1&=(s^2+s^2t)x-s^2t+t+s^2+1=0,\\ p_2&=(s^2+s^2t)y-s^2t-s+t=0,\\ p_3&=(s^2+s^2t)z-2s^2+2t+2=0. \end{aligned}$$

要求用 x 、y 、z 来表示 s 、t. 由所谓的迪克逊 (A.L.Dixon) 结式

方法 (参见 §28), 可以导出方程组

$$D\cdot\begin{pmatrix} st \\ s \\ t \\ 1 \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

其中

$$D=\begin{pmatrix} 0 & 2x+2z-2 & 6y-4x-z-2 & 6y-2x+z-4 \\ 6y-4x-z-2 & 6y-2x+z-4 & 2x-z-2 & 2x-z-2 \\ 2x-z-2 & 2x-z+4 & 2x-z-2 & 2x-z+4 \\ 2x-z-2 & 2x-z+4 & 0 & 0 \end{pmatrix}$$

称为多项式 p_1、p_2、p_3 关于变元 s、t 的 **迪克逊矩阵**. 我们把这个方程组看作为 st, s, t, 1 的线性方程组.

下面分别用克莱姆法则和高斯消去法解上述“线性”方程组.

用克莱姆法则, 可得

$$\det(D)=0,$$

$$t=\frac{\begin{vmatrix} 6y-4x-z-2 & 6y-2x+z-4 & z-2x+2 \\ 2x-z-2 & 2x-z+4 & z-2x-4 \\ 2x-z-2 & 2x-z+4 & 0 \end{vmatrix}}{\begin{vmatrix} 6y-4x-z-2 & 6y-2x+z-4 & 2x-z-2 \\ 2x-z-2 & 2x-z+4 & 2x-z-2 \\ 2x-z-2 & 2x-z+4 & 0 \end{vmatrix}},$$

$$s=\frac{\begin{vmatrix} 6y-4x-z-2 & z-2x+2 & 2x-z-2 \\ 2x-z-2 & z-2x-4 & 2x-z-2 \\ 2x-z-2 & 0 & 0 \end{vmatrix}}{\begin{vmatrix} 6y-4x-z-2 & 6y-2x+z-4 & 2x-z-2 \\ 2x-z-2 & 2x-z+4 & 2x-z-2 \\ 2x-z-2 & 2x-z+4 & 0 \end{vmatrix}}.$$

这种显式解使得我们不必去展开行列式, 就可以研究逆映射.

用高斯消去法, 可得

$$\tilde{D}\cdot\begin{pmatrix} st \\ s \\ t \\ 1 \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

其中

$$\tilde{D} = \begin{pmatrix} 2x - z - 2 & 2x - z + 4 & 0 & 0 \\ 0 & 2x + 2z - 2 & 6y - 4x - z - 2 & 6y - 2x + z - 4 \\ 0 & 0 & 2x - z - 2 & 2x - z + 4 \\ 0 & 0 & 0 & d \end{pmatrix},$$

$$d = \frac{\det(D)}{2(2x - z - 2)^2(x + z - 1)}.$$

把最后的三个方程写出来, 即得

$$\begin{aligned} &\det(D) = 0, \\ &(2x - z - 2)t + 2x - z + 4 = 0, \\ &(2x + 2z - 2)s + (6y - 4x - z - 2)t + 6y - 2x + z - 4 = 0. \end{aligned}$$

这递推地给出了所求逆映射.

同时我们也得到了曲面的隐式方程:

$$\begin{aligned} \det(D) = \quad & -4(52 + 164x - 288y + 32z + 60x^2 + 324y^2 - 3z^2 \\ & -288xy + 48xz - 72yz + 44x^3 - z^3 - 72x^2y + 24x^2z \\ & -21xz^2 + 36yz^2 - 36xyz + 4x^4 + z^4 + 4x^3z \\ & -3x^2z^2 - 2xz^3) = 0. \end{aligned}$$

就本例而言, 迪克逊结式方法导出了足够多的方程, 从而使得接着执行的两种线性求解方法直接给出了所求逆映射. 可是目前还没有一个方法能够在任何情况下都导出足够多的方程, 使得我们可以直接地得到递推解或者显式解. 在这样的情况下, 我们还不知道怎样得到方程组的显式解, 但却能间接地得到递推解, 这就是本章主要发展的一个完全方法.

§ 27 基本概念

§ 27.1 幂积的序

一个多项式方程可以写为 $p(x_1, \ldots, x_k) = 0$ 的形式, 其中 $p(x_1, \ldots, x_k)$ 是某域上的一个多项式. 方程的解也就是对应多项

式的零点．所以，我们常常不提方程而只说多项式．指定一个域 K，多项式可以看作是一些互异幂积在域 K 上的线性组合，这些幂积在域 K 上是线性无关的．比如多项式

$$p_1 = (s^2 + s^2t)x - s^2t + t + s^2 + 1,$$

在 x 的有理函数域 $\mathbf{Q}(x)$ 上，p_1 可以表示为

$$p_1 = (x-1)s^2t + (x+1)s^2 + t + 1,$$

它是变元 s 和 t 的幂积 s^2t、s^2、t、1（变元的零次幂积）的线性组合．

单变元 x 的幂积，依其幂指数的序有如下的所谓降幂序：

$$\cdots \succ x^k \succ x^{k-1} \succ \cdots \succ x \succ 1 \succ x^{-1} \succ \cdots.$$

因而单变元的多项式可以写为降幂的形式，这种写法往往给论述带来诸多便利．

类似地，对于多个变元的情形，有两种常用的序，这就是 **纯字典序** 和 **全幂序**．为着明确起见，如下给出它们的定义．

设 x_1, x_2, $\cdots$, x_k 是 k 个变元，首先指定这些变元的序，在现在这种情况下，不妨按变元下标的大小规定变元的先后顺序，表示为：

$$x = (x_1, x_2, \cdots, x_k).$$

设 n_i 是整数 $(i = 1, 2, \ldots, k)$，$n = (n_1, n_2, \cdots, n_k)$．变元 x_1, x_2, $\cdots$, x_k 的幂积可表示为

$$e = x^n = x_1^{n_1} x_2^{n_2} \cdots x_k^{n_k},$$

e 的全次数记为

$$\mathrm{tdeg}(e) = n_1 + n_2 + \cdots + n_k.$$

任意给定变元 x_1, x_2, $\cdots$, x_k 的两个幂积

$$e_1 = x_1^{r_1} x_2^{r_2} \cdots x_k^{r_k}, \quad e_2 = x_1^{s_1} x_2^{s_2} \cdots x_k^{s_k},$$

如果

$$r_i = s_i, \qquad i = 1, 2, \ldots, k$$

则称 e_1 和 e_2 是相等的, 记为 $e_1 = e_2$. 如果 e_1 和 e_2 不相等, 且

$$r_1 < s_1,\ \text{或者存在}\ j(\le k)\ \text{使得}\ r_i = s_i, i = 1, \ldots, j-1;\ r_j < s_j,$$

那么就说 e_1 按纯字典序小于 e_2, 记为 $e_1 \overset{p}{\prec} e_2$. 如果 e_1 和 e_2 不相等, 且如下三个条件之一成立:

(1) $r_1 + r_2 + \cdots + r_k < s_1 + s_2 + \cdots + s_k$;

(2) $r_1 + r_2 + \cdots + r_k = s_1 + s_2 + \cdots + s_k$, $r_1 < s_1$;

(3) $r_1 + r_2 + \cdots + r_k = s_1 + s_2 + \cdots + s_k$, $r_1 = s_1$, 并且存在 $j(\le k)$ 使得 $r_i = s_i$ 对 $i = 1, \ldots, j-1$ 成立但是 $r_j < s_j$;

那么, 就说 e_1 按全幂序小于 e_2, 记为 $e_1 \overset{t}{\prec} e_2$.

例如

$$1 \overset{p}{\prec} x_2 \overset{p}{\prec} x_2^2 \overset{p}{\prec} \cdots \overset{p}{\prec} x_1 \overset{p}{\prec} x_1x_2 \overset{p}{\prec} x_1^2.$$

$$1 \overset{t}{\prec} x_2 \overset{t}{\prec} x_1 \overset{t}{\prec} x_2 \overset{t}{\prec} x_2^2 \overset{t}{\prec} x_1x_2 \overset{t}{\prec} x_1^2 \overset{t}{\prec} \cdots.$$

我们用 $\prec$ 来表示 $\overset{p}{\prec}$ 和 $\overset{t}{\prec}$ 中的任意一个, 而用 $\preceq$ 来表示 "$\prec$ 或 $=$".

除非特别声明, 我们总假定幂积关于每个变元的次数都非负. 可是, 有时为了方便, 我们假定幂积关于每个变元的次数都可以为负, 这样做的时候, 总是把某些或全部变元取值为 0 的那些零点放到一边, 而只考虑全部变元取值都非 0 的那些零点 (即无挠零点).

§ 27.2 多项式组

域 K 上的任一多项式 $p=p(x_1,x_2,\ldots,x_k)$，可以按变元 $(x_1,x_2,\cdots,x_k)$ 的纯字典序或全幂序来写出．例如

$$p_1=(x-1)s^2t+(x+1)s^2+t+1,$$

就是按变元 (s,t) 的纯字典序（及全幂序）写出的．变元 $(x_1,x_2,\cdots,x_k)$ 的任一组互不相同的幂积在域 K 上都是线性无关的，所以，从线性的观点来看，可以把这些幂积看作是不同的变元．对于多项式方程组

$$PS:\begin{cases} p_1(x_1,x_2,\ldots,x_k)=0,\\ p_2(x_1,x_2,\ldots,x_k)=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ p_m(x_1,x_2,\ldots,x_k)=0. \end{cases}$$

我们把其中出现的关于变元 $(x_1,x_2,\cdots,x_k)$ 的所有幂积（除非特别声明，我们总是把它们记为 $e_1,e_2,\cdots,e_n$）按字典序或全幂序由大到小排列为：

$$e_n\prec\cdots\prec e_2\prec e_1.$$

这样，方程组 PS 就可以写为

$$P\cdot\begin{pmatrix} e_n\\ \vdots\\ e_2\\ e_1 \end{pmatrix}=\begin{pmatrix} 0\\ \vdots\\ 0\\ 0 \end{pmatrix}$$

其中 P 为 PS 关于幂积 $e_n,\cdots,e_2,e_1$ 的系数矩阵．我们用 p_i 和 e_j 分别作为这个矩阵的行指标和列指标，我们把方程组的这种表示称之为 **标准形式**．对应于多项式组，我们也可以作类似的表示，在此不赘述．例如在 §26 中，多项式方程组 $\{p_1=0,\ p_2=0,\ p_3=0\}$ 的迪克逊导出方程组就是用标准形式表示的．

对于一个多项式 $f=f(x_1,\cdots,x_k)$，如果其所有系数彼此是代数无关的文字，就说 f 是 **一般的**，例如

$$f_1=a_1x_1^2+a_2x_1x_2+a_3x_1+a_4x_2+a_5,$$

其中的 $a_1,\cdots,a_5$ 是代数无关的. f 中所出现幂积的全次数中最大者 d, 称为 f 的次数. 如果 f 包含所有次数小于或等于 d 的幂积, 就说 f 是 **全次的**, 例如

$$f_2 = x_1^2 + 2ax_1x_2 + (3+b)x_2^2 + 4bx_1 + 5ax_2 + 6.$$

如果 f 既是一般的又是全次的, 就说是 **一般全次的**, 例如

$$f_3 = a_1x_1^2 + a_2x_1x_2 + a_3x_2^2 + a_4x_1 + a_5x_2 + a_6.$$

设 f 关于变元 x_i 的次数为 $n_i(i=1,\cdots,k)$, 如果 f 有形式:

$$f = \sum_{i_1=1}^{n_1} \cdots \sum_{i_k=1}^{n_k} a_{(j,i_1,\cdots,i_k)} x_1^{i_1} \cdots x_k^{i_k}, \qquad 1 \le j \le k+1$$

其中所有的 a 是代数独立的, 则称 f 是 **一般完全的**.

如果我们把一般多项式的文字系数取为具体的值 (可以是任意含参数或不含参数的代数表达式), 我们就说这是一个具体 **指定**. 例如上面的 f_2 就是 f_3 的一个具体指定.

对应于多项式组的情形, 也有类似的定义.

例如考虑如下三个二变元多项式组

$$S: \begin{cases} p_1 = x^2 + axy - y + b, \\ p_2 = xy + (a+b)y - c, \\ p_3 = bx + y + 2ac, \end{cases}$$

其中 a 、b 、c 是参数. 我们说它不是"一般完全的", 因为, 比如 p_1 中 x^2 的系数是 1 而不是独立的参数, p_2 中 y 的系数 $a+b$ 代数地依赖于 p_1 的系数, 尽管 x 和 y 的最高次数分别是 2 和 1, 却不包含有幂积 x^2y 等等. 对应于这个多项式组的一般完全型是

$$G: \begin{cases} p_1' = a_{11}x^2y + a_{12}x^2 + a_{13}xy + a_{14}x + a_{15}y + a_{16}, \\ p_2' = a_{21}x^2y + a_{22}x^2 + a_{23}xy + a_{24}x + a_{25}y + a_{26}, \\ p_1' = a_{31}x^2y + a_{32}x^2 + a_{33}xy + a_{34}x + a_{35}y + a_{36}. \end{cases}$$

§ 27.3 多项式组的结式

给定 $k+1$ 个 k 变元 $(x_1,\ldots,x_k)$ 的一般多项式

$$p_0=p_0(x_1,\ldots,x_k),\ \cdots,\ p_k=p_k(x_1,x_2,\ldots,x_k).$$

用 c_{ij} 表示它们的系数，则 $p_0,\ldots,p_k$ 的结式定义为 c_{ij} 的一个不可约多项式 $\mathrm{res}(p_0,\ldots,p_k)$，且对于 $p_0,\ldots,p_k$ 的任一具体指定 $\bar{p}_0,\ldots,\bar{p}_k$，只要 $\bar{p}_0,\ldots,\bar{p}_k$ 有一个公共零点，就有 $\mathrm{res}(\bar{p}_0,\ldots,\bar{p}_k)=0$. 不失一般性，我们只考虑无挠零点.

一般来说，由 $p_0,\ldots,p_k$ 得到如下形式的方程组是不困难的：

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix}.$$

其中 $a_{ij}\in K$, f_i, g_j 都是 $x_1,\ldots,x_k$ 的多项式，且在 $p_0,\ldots,p_k$ 的每一个无挠零点处，$f_1,\ldots,f_n$ 不都等于 0，而 $g_1,\ldots,g_n$ 都等于 0. 令

$$\mathrm{Res}(p_0,\ldots,p_k)=\begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix},$$

则根据线性代数方程组的理论易知，结式 $\mathrm{res}(p_0,\ldots,p_k)$ 必然是 $\mathrm{Res}(p_0,\ldots,p_k)$ 的一个因子. 如果可以证明 $\mathrm{Res}(p_0,\ldots,p_k)$ 和 $\mathrm{res}(p_0,\ldots,p_k)$ 有相同的次数，那么 $\mathrm{Res}(p_0,\ldots,p_k)$ 事实上就是多项式组 $p_0,\ldots,p_k$ 的结式.

§ 28 迪克逊导出方程组

由 §26 的例子可以看出，借助于线性方法来求解一般的非线性方程组，其第一步也是关键的一步是从原方程组得到导出方程

组. 本节介绍迪克逊结式方法, 它正是我们在 §26 例子中用以产生导出方程组的方法.

早在 1779 年, 贝佐就已发展了一个计算两个单变元多项式结式的方法. 它是用较低阶的行列式来表示结式的. 这就是著名的贝佐结式 (参见本书第 **2** 章). 贝佐结式在多变元情形可以从两个方向来发展, 其中之一对应于所谓迪克逊结式. 正是本节和下一节的主题.

首先让我们描述凯莱 (A.Cayley) 构造贝佐结式的方法[24].

给定两个次数为 n 的单变元多项式 $f(x)$ 和 $g(x)$, 设 α 是异于 x 的变元, 则行列式

$$\Delta(x,\alpha)=\begin{vmatrix} f(x) & g(x) \\ f(\alpha) & g(\alpha) \end{vmatrix}$$

是关于变元 x、α 的多项式, 并且当 $x=\alpha$ 时 $\Delta(x,\alpha)=0$, 这意味着 $x-\alpha$ 是 $\Delta(x,\alpha)$ 的因子. 这样, 多项式

$$\delta(x,\alpha)=\frac{\Delta(x,\alpha)}{x-\alpha}$$

就是一个关于变元 α 的 $n-1$ 次多项式, 并且关于 x 和 α 是对称的. 可以把多项式 $\delta(x,\alpha)$ 写为

$$\delta(x,\alpha)=c_1(x)\alpha^{n-1}+c_2(x)\alpha^{n-2}+\cdots+c_n(x).$$

其中 $c_i(x)$ 是关于变元 x 的多项式且次数小于或等于 $n-1$.

设若 x_0 是多项式 $f(x)$ 和 $g(x)$ 的一个公根, 则显见在 $x=x_0$ 处多项式 $\delta(x,\alpha)$ 等于 0, 即 $\delta(x_0,\alpha)=0$, 而不论 α 取什么值. 所以, 在 $x=x_0$ 处, 多项式 $\delta(x,\alpha)$ 关于变元 α 的各阶幂积的系数都等于 0, 即

$$\{c_1(x_0)=0,\ c_2(x_0)=0,\ \ldots,\ c_n(x_0)=0\}.$$

一般地, 多项式 $f(x)$ 和 $g(x)$ 的任一公根必然是多项式方程组

$$\{c_1(x)=0,\ c_2(x)=0,\ \ldots,\ c_n(x)=0\}$$

的解. 我们把这个方程组看作是 x 的各不同幂积 $x^{n-1}, x^{n-2}, \ldots, x, x^0=1$ 的线性方程组 (n 个方程 n 个变元). 如其有解则必然是非零解 (因 $x^0=1$). 故其系数矩阵的行列式为 0, 这个系数矩阵就是多项式 $f(x)$ 和 $g(x)$ 的贝佐矩阵, 而其行列式即是贝佐结式.

1908 年, 迪克逊把凯莱的方法推广到三个二变元多项式的情形[13]. 这个方法极易推广到 $k+1$ 个 k 变元多项式的情形. 下面就具体地指定的多项式组来介绍这个方法, 一般情形的构造方法与此类似, 我们将在下一节讨论. 一般情形与具体指定情形在算法上的这种一致性, 不仅迪克逊结式如此, 而是广泛地成立的. 不过在其他情况时, 我们将不重复这种讨论而代之以同样广泛适应的其他方法.

给定 $k+1$ 个 k 变元多项式组

$$PS: \begin{cases} p_1(x_1, x_2, \ldots, x_k), \\ p_2(x_1, x_2, \ldots, x_k), \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ p_{k+1}(x_1, x_2, \ldots, x_k). \end{cases}$$

设 $\alpha_1, \alpha_2, \ldots, \alpha_k$ 是 k 个新变元, 则如下行列式

$$\Delta(x_1, \cdots, x_k, \alpha_1, \cdots, \alpha_k) =$$

$$\begin{vmatrix} p_1(x_1, x_2, \cdots, x_k) & \cdots & p_{k+1}(x_1, x_2, \cdots, x_k) \\ p_1(\alpha_1, x_2, \cdots, x_k) & \cdots & p_{k+1}(\alpha_1, x_2, \cdots, x_k) \\ p_1(\alpha_1, \alpha_2, \cdots, x_k) & \cdots & p_{k+1}(\alpha_1, \alpha_2, \cdots, x_k) \\ \cdots & \cdots & \cdots \\ p_1(\alpha_1, \alpha_2, \cdots, \alpha_k) & \cdots & p_{k+1}(\alpha_1, \alpha_2, \cdots, \alpha_k) \end{vmatrix}$$

是关于变元 x_i 、α_i 的多项式 $(i=1,2,\ldots,k)$, 并且当 $x_i=\alpha_i$ $(i=1,2,\ldots,k)$ 时

$$\Delta(x_1, \cdots, x_k, \alpha_1, \cdots, \alpha_k) = 0.$$

这意味着 $(x_1-\alpha_1)\cdot(x_2-\alpha_2)\cdots(x_k-\alpha_k)$ 是 $\Delta(x_1, \cdots, x_k, \alpha_1, \cdots, \alpha_k)$ 的因子. 令

$$\delta(x_1, \cdots, x_k, \alpha_1, \cdots, \alpha_k) = \frac{\Delta(x_1, \cdots, x_k, \alpha_1, \cdots, \alpha_k)}{(x_1-\alpha_1)\cdots(x_k-\alpha_k)}.$$

这是一个多项式, 称之为 $p_1, p_2, \ldots, p_{k+1}$ 的 **迪克逊多项式**. 不妨把这个多项式看作为 $\alpha_1, \cdots, \alpha_k$ 的多项式, 并把 $\alpha_1, \cdots, \alpha_k$ 的各不同幂积之系数记之为

$$c_i(x_1, x_2, \ldots, x_k). \qquad i = 1, 2, \ldots, n$$

它们都是关于变元 $x_1, x_2, \ldots, x_k$ 的多项式. 我们称这些 c_i 为 $p_1, p_2, \ldots, p_{k+1}$ 的 **迪克逊导出多项式组**. 对应于方程组的情形, 我们称方程组

$$c_i(x_1, x_2, \ldots, x_k) = 0 \qquad i = 1, 2, \ldots, n$$

为方程组 $p_1 = 0, p_2 = 0, \ldots, p_{k+1} = 0$ 的 **迪克逊导出方程组**. 容易知道, 原方程组的解必定是其迪克逊导出方程组的解.

在上述迪克逊导出方程组中, 我们把其中出现的关于变元 $(x_1, x_2, \cdots, x_k)$ 的所有幂积按字典序 (或全幂序) 由大到小写为

$$e_n, \quad \cdots, \quad e_2, \quad e_1.$$

这样, 迪克逊导出方程组就可以写为标准形式:

$$D \cdot \begin{pmatrix} e_n \\ \vdots \\ e_2 \\ e_1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

其中 D 为迪克逊导出方程组关于幂积 $e_n, \cdots, e_2, e_1$ 的系数矩阵, 称为多项式组 $p_1, p_2, \ldots, p_{k+1}$ 的 **迪克逊矩阵**.

作为一个例子, 现在来解决 §26 遗留下来的一个问题, 即构造多项式组

$$\begin{aligned} p_1(s,t) &= (s^2 + s^2 t)x - s^2 t + t + s^2 + 1, \\ p_2(s,t) &= (s^2 + s^2 t)y - s^2 t - s + t, \\ p_3(s,t) &= (s^2 + s^2 t)z - 2s^2 + 2t + 2 \end{aligned}$$

的迪克逊导出多项式组.

由计算, 可得

$$\Delta(s,t)=\begin{vmatrix} p_1(s,t) & p_2(s,t) & p_3(s,t) \\ p_1(\alpha,t) & p_2(\alpha,t) & p_3(\alpha,t) \\ p_1(\alpha,\beta) & p_2(\alpha,\beta) & p_3(\alpha,\beta) \end{vmatrix}$$

$$=\begin{vmatrix} p_1(s,t)-p_1(\alpha,t) & p_2(s,t)-p_2(\alpha,t) & p_3(s,t)-p_3(\alpha,t) \\ p_1(\alpha,t)-p_1(\alpha,\beta) & p_2(\alpha,t)-p_2(\alpha,\beta) & p_3(\alpha,t)-p_3(\alpha,\beta) \\ p_1(\alpha,\beta) & p_2(\alpha,\beta) & p_3(\alpha,\beta) \end{vmatrix}$$

$$=(s-\alpha)(t-\beta)(c_1\alpha^3+c_2\alpha^2+c_3\alpha+c_4),$$

$$\delta(s,t)=\Delta(s,t)/((s-\alpha)(t-\beta))=c_1\alpha^3+c_2\alpha^2+c_3\alpha+c_4,$$

其中

$$\begin{aligned} c_1 &= (2x+2z-2)s+(6y-4x-z-2)t+6y-2x+z-4, \\ c_2 &= (6y-4x-z-2)st+(6y-2x+z-4)s \\ &\quad +(2x-z-2)t+2x-z-2, \\ c_3 &= (2x-z-2)st+(2x-z+4)s+(2x-z-2)t+2x-z+4, \\ c_4 &= (2x-z-2)st+(2x-z+4)s. \end{aligned}$$

即为 p_1, p_2, p_3 的迪克逊导出多项式组.

§ 29 一般情形的迪克逊结式

给定一般多项式组

$$PS:\begin{cases} p_1=p_1(x_1,x_2,\ldots,x_k), \\ p_2=p_2(x_1,x_2,\ldots,x_k), \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ p_{k+1}=p_{k+1}(x_1,x_2,\ldots,x_k). \end{cases}$$

对于 $1\le i\le k$, 设

$$d_{\max_i}=\max\{\deg(p_1,x_i),\deg(p_2,x_i),\ldots,\deg(p_{n+1},x_i)\}.$$

沿用 §28 的记法并作类似的讨论, 易知 $p_1, p_2, \ldots, p_{k+1}$ 的迪克逊多项式

$$\delta = \delta(x_1, \cdots, x_k, \alpha_1, \cdots, \alpha_k)$$

关于 α_i 和 $x_i (1 \le i \le k)$ 的次数分别为

$$\deg(\delta, \alpha_i) = (k+1-i)d_{\max_i} - 1, \quad \deg(\delta, x_i) = i \cdot d_{\max_i} - 1.$$

原多项式组 $p_1, p_2, \ldots, p_{k+1}$ 的任一零点必然是其迪克逊多项式的零点, 而不论 $\alpha_1, \cdots, \alpha_k$ 取什么值, 所以也必然是其导出多项式组

$$c_i(x_1, x_2, \ldots, x_k) \qquad i = 1, 2, \ldots, n$$

的零点. 这里,

$$n = \prod_{i=1}^{k}((k+1-i)d_{\max_i}) = k! \prod_{i=1}^{k} d_{\max_i}.$$

并且, 在所有 $c_i(x_1, x_2, \ldots, x_k)$ 中, 变元 $x_1, \cdots, x_k$ 的各不同幂积的个数为

$$\prod_{i=1}^{k}(i \cdot d_{\max_i}) = k! \prod_{i=1}^{k} d_{\max_i} = n.$$

我们用新变元 $e_i (i = 1, \ldots, n)$ 依全幂序来表示这些幂积 (其中有些可能并不出现):

$$e_1 = 1, \quad e_2 = x_k, \quad \cdots, \quad e_n = \prod_{i=1}^{k} x_i^{i \cdot d_{\max_i} - 1}.$$

把导出多项式组看作是 $e_1, e_2, \cdots, e_n$ 的线性式, 其系数矩阵称为 $PS: \{p_1, p_2, \ldots, p_{k+1}\}$ 的 **迪克逊矩阵**, 记之为 D. 迪克逊矩阵的行列式称为 $PS: \{p_1, p_2, \ldots, p_{k+1}\}$ 的 **迪克逊结式**.

这样，方程组 $\{p_1 = 0, p_2 = 0, \ldots, p_{k+1} = 0\}$ 就蕴含线性方程组

$$D \cdot \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

换言之，多项式组的迪克逊结式为 0 是该多项式组有公共零点的必要条件.

例如，考虑由两个双二次式和一个线性式组成的系统：

$$\begin{aligned} f_1 &= a_1 x^2 y^2 + a_2 x_1^2, \\ f_2 &= b_1 x^2 y^2 + b_2 y^2, \\ f_3 &= u_1 x_1 + u_2 x_2 + u_3. \end{aligned}$$

它的迪克逊多项式是

$$(\alpha^3\beta, \alpha^3, \alpha^2\beta, \alpha^2, \alpha\beta, \alpha, \beta, 1) \cdot D \cdot (xy^3, xy^2, y^3, xy, y^2, x, y, 1)^{Tr}.$$

其中 $D =$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & a_2b_1u_1 & a_2b_1u_2 & a_2b_1u_3 \\ 0 & 0 & 0 & a_2b_1u_1 & a_2b_1u_2 & 0 & a_2b_1u_3 & 0 \\ 0 & 0 & 0 & a_2b_1u_2 & a_1b_2u_1 & a_2b_1u_3 & 0 & 0 \\ 0 & a_2b_1u_2 & 0 & a_2b_1u_3 & a_1b_2u_1 & 0 & 0 & 0 \\ 0 & a_1b_2u_1 & a_1b_2u_2 & 0 & a_1b_2u_3 & a_2b_2u_1 & a_2b_2u_2 & a_2b_2u_3 \\ a_1b_2u_1 & 0 & a_1b_2u_3 & a_2b_2u_1 & 0 & 0 & a_2b_2u_3 & 0 \\ a_1b_2u_2 & a_1b_2u_3 & 0 & a_2b_2u_2 & 0 & a_2b_2u_3 & 0 & 0 \\ a_1b_2u_3 & 0 & 0 & a_2b_2u_3 & 0 & 0 & 0 & 0 \end{pmatrix}$$

是多项式组 $\{f_1, f_2, f_3\}$ 的迪克逊矩阵，其行列式即为迪克逊结式：

$$\begin{aligned} \det(D) = \ & a_1^2 a_2^4 b_1^2 b_2^4 u_3^4 (a_1^2 b_1^2 u_3^4 + 2a_1^2 b_1 b_2 u_1^2 u_3^2 + a_1^2 b_2^2 u_1^4 \\ & -2a_1 a_2 b_1^2 u_2^2 u_3^2 - 2a_1 a_2 b_1 b_2 u_1^2 u_2^2 + a_2^2 b_1^2 u_2^4). \end{aligned}$$

对于 $k+1$ 个 k 变元的一般完全多项式，迪克逊证明：迪克逊结式为 0 是它们有仿射零点的一个必要条件[13].

§ 30 显 式 解

给定一个多项式组

$$PS: \quad \begin{cases} p_1 = p_1(x_1, x_2, \cdots, x_k), \\ p_2 = p_2(x_1, x_2, \cdots, x_k), \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ p_k = p_k(x_1, x_2, \cdots, x_k). \end{cases}$$

设

$$DPS: \quad \begin{cases} q_1 = c_{1n}e_n + \cdots + c_{12}e_2 + c_{11}e_1, \\ q_2 = c_{2n}e_n + \cdots + c_{32}e_2 + c_{21}e_1, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ q_m = c_{mn}e_n + \cdots + c_{m2}e_2 + c_{m1}e_1. \end{cases}$$

其中 $e_i(i = 1, 2, \cdots, n)$ 为 $(x_2, \cdots, x_k)$ 的幂积, 且

$$e_n \prec \cdots \prec e_2 \prec e_1;$$

而 $c_{ij} \in K[x_1]$ $(i = 1, 2, \cdots, m; j = 1, 2, \cdots, n)$, 即 c_{ij} 是 x_1 的多项式. 当我们说多项式组 DPS 是 PS 的迪克逊导出多项式组时, 如 §28 和 §29 所述, 它是唯一确定的. 由 PS 还可得到另外一些有确切含义的导出多项式组 (如麦考莱 (F.S.Macaulay) 导出多项式组等, 参见 §32), 一般来说它们是互不相同的, 所以"导出多项式组"是一个含义颇广泛的术语. 我们说 DPS 是 PS 的一个导出多项式组, 其基本含义是 PS 的每个 (无挠) 零点必定是 DPS 的 (无挠) 零点. 为了使得 DPS 不是徒具形式, 在形式上要求 $n - m \leq l - k$. 这里 l 表示 PS 中所出现的不同幂积的个数. 我们期望得到的导出多项式组, 在形式上最好是 $n = m$, 即导出多项式组 (对应于方程组的情形是导出方程组) 中的多项式的个数等于其中所含不同幂积的个数.

本书 §26 的例子正好是我们所期望的情形. 在那里, 方程组的解既可以用递推式表示, 也可以用显式表示. 在很多情况下, 方程组的解确实可以用显式来表示.

令 DPS 是 PS 的一个导出方程组 (或导出多项式组),M 是其标准形式的系数矩阵. 设 M 的秩为 $r = \mathrm{rank}(M)$, 并设 $M_j(j \le r)$ 是 M 的一个秩为 j 的子矩阵, 它由 M 的某 j 行组成. 对于 M 的任一子矩阵 (或任一子式), 我们总是根据它在 M 中的位置用相应的幂积 e_i 作为其列指标. M_j 的列指标为 e_i 的列向量记为 C_{ji}, $i = 1,2,\cdots,n$; τ_j 表示 M_j 的一个 j 阶非奇异子式, 有时为了明确, 也把相应的列指标 $e'_1, \cdots, e'_r$ 写出, 记为 $\tau_j(e'_1,\cdots,e'_r)$; $\tau_j^{[e_s,e_t]}$ 表示在 τ_j 中用 M_j 的列 C_{jt} 取代 τ_j 的列 C_{js} 后所得的行列式.

[例] 设某多项式组的导出多项式组为

$$\begin{aligned} q_1 &= a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6, \\ q_2 &= b_1x^2 + b_2xy + b_3y^2 + b_4x + b_5y + b_6, \\ q_3 &= c_1x^2 - c_2xy + c_3y^2 + c_4x + c_5y + c_6. \end{aligned}$$

则

$$M_3 = M = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{pmatrix},$$

$$\tau_3 = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}, \quad \tau_3^{[x^2,x]} = \begin{vmatrix} a_4 & a_2 & a_3 \\ b_4 & b_2 & b_3 \\ c_4 & c_2 & c_3 \end{vmatrix},$$

$$\tau_3^{[x^2,y]} = \begin{vmatrix} a_5 & a_2 & a_3 \\ b_5 & b_2 & b_3 \\ c_5 & c_2 & c_3 \end{vmatrix}, \quad \tau_3^{[x^2,1]} = \begin{vmatrix} a_6 & a_2 & a_3 \\ b_6 & b_2 & b_3 \\ c_6 & c_2 & c_3 \end{vmatrix}.$$

定理 1[24] 若 τ_j 的列指标为 $e'_1, \cdots, e'_j$, 即

$$\tau_j(e'_1, \cdots, e'_j) \neq 0,$$

则显然 M_j 中列指标为 $e'_1, \cdots, e'_j$ 的 j 个列向量

$$V: \quad \{C'_{j1}, \quad \cdots, \quad C'_{jj}\}$$

线性无关；如果用 M_j 的其余 $n-r$ 个列向量中的每一个替换 V 中的任一指定的向量，比如说 C_1'，都得到线性相关的向量组，那么

$$\tau_j(e_1', \cdots, e_j')$$

为零是原多项式组有公共的无挠零点的一个必要条件.

定理 2 对于 $k+1$ 个 k 变元多项式组，沿用上述记号，我们有：设若 M_j 是 $j\times(j+1)$ 矩阵，且包含有列指标 $x_1, x_2, \cdots, x_k, 1$，如果

$$\tau = \tau_j(e_{j+1}, \cdots, e_{k+2}, x_1, x_2, \cdots, x_k) \neq 0,$$

那么，原多项式组或者没有公共零点，或者其公共零点为一点：

$$\left(-\frac{\tau^{[x_1,1]}}{\tau}, -\frac{\tau^{[x_2,1]}}{\tau}, \cdots, -\frac{\tau^{[x_k,1]}}{\tau}\right).$$

上面两个论断适应于任何导出方程组，是线性代数方程组理论的简单推论.

§31 聚 筛 法

给定一个多项式方程组

$$PS: \quad \begin{cases} p_1 = p_1(x_1, x_2, \cdots, x_k) = 0, \\ p_2 = p_2(x_1, x_2, \cdots, x_k) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ p_k = p_k(x_1, x_2, \cdots, x_k) = 0. \end{cases}$$

我们可以"聚"得它的一个导出方程组 DPS(比如迪克逊导出方程组). 按照定义，DPS 的无挠零点集包含 PS 的无挠零点集，但反之则不一定对. 怎么样才能把那些多余的零点给"筛"掉呢？ §22 给出的" WR 相对分解算法"可以帮助我们做到. 这种求解方程组时先"兼收并蓄"而后"去伪存真"的方法，就是所谓的"聚筛法".

沿用 §30 的记号. 给定 PS 如上. 首先作如下几步：

步骤 1 保留 x_1 作为参数，构造 PS 的 k 个方程关于变元 $x_2,\cdots,x_k$ 的一个导出方程组 DPS.

步骤 2 把 DPS 表示为标准形式 (x_1 仍作为参数):

$$DPS: \begin{cases} q_1 = c_{1n}e_n + \cdots + c_{12}e_2 + c_{11}e_1 = 0, \\ q_2 = c_{2n}e_n + \cdots + c_{22}e_2 + c_{21}e_1 = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ q_m = c_{mn}e_n + \cdots + c_{m2}e_2 + c_{m1}e_1 = 0. \end{cases}$$

步骤 3 对 $e_n,\cdots,e_2,e_1$ 的线性方程组 DPS 施行不带分式的高斯消元，从而得到：

$$GPS: \begin{cases} h_n = b_{nn}e_n + \cdots + b_{n2}e_2 + b_{n1}e_1 = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ h_2 = b_{22}e_2 + b_{21}e_1 = 0, \\ h_1 = b_{11}e_1 = 0. \end{cases}$$

其中 $b_{ij} \in K[x_1]$ $(i = 1,2,\cdots,m; j = 1,2,\cdots,n)$.

回头重新把 x_1 看作未知元，这样 GPS 就可写为：

$$GPS: \begin{cases} h_1 = h_1(x_1,x_2,\ldots,x_k) = 0, \\ h_2 = h_2(x_1,x_2,\ldots,x_k) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ h_m = h_m(x_1,x_2,\ldots,x_k) = 0. \end{cases}$$

这前 3 步 (步骤 1 至步骤 3) 合起来叫做 **GPS 算法**. 因为它是获取方程组 GPS 的步骤. 为此而编的通用程序叫做 **GPS程序**.

现在让我们先来考虑一种较为简单的情形：

步骤 4 (如果可能) 从 GPS 的 m 个方程 $\{h_1 = 0,\ h_2 = 0,\ \cdots,\ h_m = 0\}$ 中选出 k 个，使之形成为关于变元 $\{x_1,\cdots,x_k\}$ 的一个三角型方程组 TS; 相应的三角列记为：

$$TS: \begin{cases} f_1(x_1), \\ f_2(x_1,x_2), \\ \cdots\cdots\cdots \\ f_s(x_1,x_2,\ldots,x_k). \end{cases}$$

步骤 5 把 TS 分解为有限个正常升列 (参见 §22 定理 1).

步骤 6 对每个正常升列 AC 相对于多项式组 $\{p_1,p_2,\cdots,p_k\}$ 作单纯分解.

这样, 得到的每一个与多项式组 $\{p_1,p_2,\cdots,p_k\}$ 整相关的正常升列就表示原方程组 $\{p_1=0,p_2=0,\cdots,p_k=0\}$ 的一组解；所有的这些解合在一起, 就是 $\{p_1=0,p_2=0,\cdots,p_k=0\}$ 的全部 (无挠) 解.

在上述步骤 4 中, 如果找到的不是恰好由 k 个多项式组成的关于所有变元 $x_1,x_2,\ldots,x_s$ 的三角列, 而是由 j 个多项式组成的关于 j 个变元 (比如说 $x_1,\ldots,x_j$) 的三角列, 那么把 $x_{j+1},\cdots,x_k$ 看作 $u_1,\cdots,u_r$, 利用 §25 的方法, 可以求得方程组

$$GS:\begin{cases} f_1=f_1(x_{j+1},\ldots,x_k;x_1)=0,\\ f_2=f_2(x_{j+1},\ldots,x_k;x_1,x_2)=0,\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ f_s=f_j(x_{j+1},\ldots,x_k;x_1,x_2,\ldots,x_j)=0,\\ g=g(x_{j+1},\cdots,x_k,x_1,\cdots,x_j)=0,\end{cases}\qquad (g\in PS)$$

亦即原方程组 PS 的解 (因为上面每一步都没有漏掉无挠解, 同时最后一步又使得解没有增加).

我们把上述解方程组的方法称为**聚筛法**, 或 **Gather-and-Sift 算法**[38], 而 GPS 算法只是它的前半部分.

[例 1] (三角形三角平分线与三边之长度关系) 设 ABC 是一个三角形, a、b 和 c 分别是角 A、B 和 C 的对边长度, a_i、a_e 分别是角 A 的内、外角平分线的长度, b_e 是角 B 的外角平分线的长度. 试用 a_i、a_e 和 b_e 来表示 a、b 和 c.

这个问题可以 (用角平分线的计算公式) 归结为求解如下多项式方程组[19]:

$$\begin{cases} p_1=a_i^2(b+c)^2-cb(c+b-a)(c+b+a)=0,\\ p_2=a_e^2(c-b)^2-cb(a+b-c)(c-b+a)=0,\\ p_3=b_e^2(c-a)^2-ac(a+b-c)(c+b-a)=0,\end{cases}\qquad(31.1)$$

其中 a、b、c 是未知元, a_i、a_e、b_e 是参变元.

作多项式组 $\{p_1,p_2,p_3\}$ 关于变元 $\{b,c\}$ 的迪克逊导出多项式组, 并把它按全幂序 (降幂排列) 表示为标准形式, 然后做不带分式的高斯消元得到 GPS. 我们所编制的程序 **GPS** 给出的 GPS 由 13 个多项式组成 : $\{h_1,\cdots,h_{13}\}$. 很容易从 GPS 得到一个升列 (它由 GPS 的最后三个多项式组成):$\{h_{13},h_{12},h_{11}\}$, 即

$$
\begin{aligned}
&f_1 = h_{13} = P_1(a_i,a_e,b_e,a) = 0,\\
&f_2 = h_{12} = Q_1(a_i,a_e,b_e,a)b + Q_2(a_i,a_e,b_e,a) = 0,\\
&f_3 = h_{11} = R_1(a_i,a_e,b_e,a,b)c + R_2(a_i,a_e,b_e,a,b) = 0.
\end{aligned}
$$

其中 R_1,R_2,Q_1,Q_2 和 P_1 分别是有 162, 390, 298, 162 和 330 项的多项式. 由于 f_1 是一个不可约的多项式, f_2 是关于 b 的线性式而 f_3 是关于 c 的线性式, 所以, 升列 $\{f_1,f_2,f_3\}$ 是不可约的, 因而不需要进一步验证就知 $\{P_1,P_2,P_3\}$ 恰好就是所求.

求解该问题的上述程序运行时间 158 秒 (PC 586/75).

这一问题的解首先是由高和王[19]给出的. 他们采取伪除法和结式计算交替使用, 加以某些人工干预, 用 Lisp 语言在一台 Sun 4 工作站上实现. 为获得第一个方程花费了 19 个机时. 最近他们又用 Maple 在 Sun Sparc-10 上实现, 据报告运行速度在其原有基础上提高约一个数量级 (见 [19] 脚注).

其实这问题还可以解得更快, 只要我们将变元的排序改为 $\{c,b,a\}$, 也就是说, 采用多项式组 $\{p_1,p_2,p_3\}$ 关于变元 $\{b,a\}$ 的迪克逊导出多项式组, 其余步骤不变. 这样, 最后得到的升列中, 第一个方程仅依赖于 c, 第二个方程线性地依赖于 b 而不依赖于 a, 第三个方程线性地依赖于 a. 与此相应的程序在 PC 586/120 上的运行时间仅为 19 秒! 这与最初的 19 个机时相比, 效率的提高是显著的.

企图计算系统 (31.1) 的格罗布讷基或麦考莱商 (见 §32) 的尝试, 各在 Sun Sparc-10 上运行一整天后未获结果[24]. 看来目前常用的这些方法中, 至少就解这类问题而言, "聚筛法" 可能是最优的.

当然在一般情况下，由步骤 4 得到的三角列是可约的，因而进一步"筛"去多余零点的工作是必须的.

[例 2] (神经网络问题).

$$PS: \quad \begin{cases} 1 - cx + xy^2 + xz^2 = 0, \\ 1 - cy + yx^2 + yz^2 = 0, \\ 1 - cz + zx^2 + zy^2 = 0, \end{cases}$$

其中 x, y, z 是未知元，c 是参数.

使用 GPS 算法 (即做步骤 1，步骤 2，步骤 3)，我们得到

$$GPS = \{h_1, h_2, \cdots, h_7\},$$

其中

$$\begin{aligned} h_1 &= x(1 - cz + zx^2 + zy^2), \\ h_4 &= x^3(x - y)(x^2y + xy^2 + 1), \\ h_7 &= x^2(-2x^3 + cx - 1)(2cx^4 - 2x^3 - c^2x^2 - 2cx - 1) \\ &\quad \cdot(x^3 - cx - 1)^2(2x^4 - 3cx^2 + x + c^2)^2. \end{aligned}$$

我们只考虑无挠零点，所以可以把因子 x 从各多项式中约去 (事实上，容易看出 $x = 0$ 不是原方程组的解)，从而得到 $AC = \{P_1, P_2, P_3\}$，其中

$$\begin{aligned} P_1 &= (-2x^3 + cx - 1)(2cx^4 - 2x^3 - c^2x^2 - 2cx - 1) \\ &\quad \cdot(x^3 - cx - 1)(2x^4 - 3cx^2 + x + c^2), \\ P_2 &= (x - y)(x^2y + xy^2 + 1), \\ P_3 &= 1 - cz + zx^2 + zy^2. \end{aligned}$$

多项式组 AC 的零点集包含了 PS 的所有零点，现在我们要把多余出来的零点给筛掉.

由于 P_1 和 P_2 显然分别有 4 个和 2 个因子，所以 AC 可以容易地被分解为 8 个分支： $AC_1, \cdots, AC_8$. 应用 WR 分解程序作每一个 AC_i $(i = 1, \cdots, 8)$ 相对于多项式组 PS 的单纯分解，得到 5 个正常升列： $CS_1, \cdots, CS_5$，它们与多项式组 PS 是整相关的，即

$S = \{CS_1, \cdots, CS_5\}$ 是方程组 PS 的所有参数解．其中

$$
\begin{aligned}
CS_1 &= \{-2x^3 + cx - 1, x - y, 1 - cz + zx^2 + zy^2\},\\
CS_2 &= \{2x^4 - 3cx^2 + x + c^2, x - y, 1 - cz + zx^2 + zy^2\},\\
CS_3 &= \{x^3 - cx - 1, x^2y + xy^2 + 1, 1 - cz + zx^2 + zy^2\},\\
CS_4 &= \{2x^4 - 3cx^2 + x + c^2, y(2c^4x^3 - 156cx^3 + 76c^3x^2\\
&\quad +36x^2 - 6c^2x - 2c^5x - 42c^4) + 8c^3x^3 - 36x^3 + c^5x^2\\
&\quad +84c^2x^2 - 7c^4x - 42cx - c^6 - 42c^3, 1 - cz + zx^2 + zy^2\},\\
CS_5 &= \{2cx^4 - 2x^3 - c^2x^2 - 2cx - 1, y(-2c^{12}x^3 + 92c^9x^3\\
&\quad -416c^6x^3 - 576c^3x^3 - 64x^3 + 100c^8x^2 - \cdots)\\
&\quad +2c^{11}x^3 + 32c^8x^3 - 384c^5x^3 - 64c^2x^3 - c^{13}x^2\\
&\quad +48c^{10}x^2 - \cdots, 1 - cz + zx^2 + zy^2\}.
\end{aligned}
$$

§ 32 聚筛法一例：WRSOLVE

许多方法都可纳入聚筛法的框架，关键是谁的效率高．而实际情况往往是，某种方法对某类问题较为有效，不可以笼统地评价．本节将要介绍的方法可以看作是聚筛法的一种极端情况．它不用高维结式，但也是一种完全的算法，效率高，且无须人工干预．

给定一代数多项式方程组

$$
PS: \begin{cases}
p_1 = p_1(x_1, x_2, \ldots, x_k) = 0,\\
p_2 = p_2(x_1, x_2, \ldots, x_k) = 0,\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
p_k = p_k(x_1, x_2, \ldots, x_k) = 0.
\end{cases}
$$

基于 §22 给出的 WR 相对分解算法，我们设计出一种新的有效算法，将一般代数方程组 PS 化成一系列三角型方程组 AS_i，使得

$$
\mathrm{Zero}(PS) \setminus \mathrm{Zero}(CS) = \bigcup(\mathrm{Zero}(AS_i) \setminus \mathrm{Zero}(CS_i))
$$

这里 CS 和 CS_i 是一系列非退化条件．

从本质上讲，代数方程组的相对分解和整序是密不可分的，基于任何一种相对分解，我们都可以得到一种整序的方法．因为，整序

中的每一次消元过程实际上就是关于已有升列的相对分解. 本节叙述的 **WRSOLVE** 算法正是基于这样一种思想，先用 **WR** 算法整序，然后类似上节所述"筛"的过程用 **WR** 算法去掉多余的零点. 由于 **WR** 算法不需要因式分解. 这就保证了 **WRSOLVE** 算法在遇到方程组分解时具有较高的效率. 此外，该算法还有另一显著特点，在非退化条件的处理上不同于吴氏方法，我们采用 §24 中的弱非退化条件，极大地降低了算法的复杂度. 总之，**WRSOLVE** 算法是一个完全的求解非线性代数方程的方法，我们已编出 400 余行的 **MAPLE** 通用程序，见附录 C. 下面我们将详细叙述 **WRSOLVE** 算法，并通过运算实例说明 **WRSOLVE** 是一种新的求解非线性代数方程的有力工具.

本算法的"筛"的部分 (即 WR 算法) 在 §22 已经详细介绍. 这里主要给出整序过程的算法. 在叙述算法之前，我们引入代数方程组的最大升列的概念.

设 AS_1 和 AS_2 是 PS 的两个子升列，定义偏序 $AS_1 \prec AS_2$ 如下：

1. AS_2 中方程的数目不小于 AS_1 中方程的数目.

2. 如果方程数相等，AS_2 中每个方程关于其导元的最高次数之和不大于 AS_1 中每个方程关于其导元的最高次数之和.

因此 PS 的最大升列 AS 是指满足上面偏序的最大值.

WRSOLVE 算法步骤：

输入: PS，一个方程组； VAR，一组变量的集合.

输出: $\{AS_1, AS_2, \cdots\}$，三角型方程组的集合.

初始化: $Pset = \{PS\}; Aset = \{\ \}$.

Step 1 如果 $Pset$ 是空集，转到第 7 步. 否则从 $Pset$ 取出任一方程组 PS.

Step 2 根据 VAR 计算 PS 的极大升列 AS. 如果 $PS \setminus AS$ 是空集，把 AS 并入集合 $Aset$ 中，并把 PS 从集合 $Pset$ 中消去，返

回第 1 步. 否则从集合 $PS \setminus AS$ 中取出一多项式 g, 使得 g 关于 AS 的所有导元的次数和最小.

Step 3 判断 AS 是否为正常升列. 如果为真, 转到第 4 步. 否则在 $AS = [f_1, f_2, \cdots, f_s]$ 中有一最小的正整数 $k > 1$, 使得 f_k 的导系数 $\bar{g}$ 关于子升列 $\overline{AS} = [f_1, f_2, \cdots, f_{k-1}]$ 的结式为零而伪余式不为零. 然后计算 $\overline{AS}$ 关于 $\bar{g}$ 的 **WR** 分解, 得到一系列新的正常升列 $\overline{AS}_i$. 令 $PS_i = AS_i \bigcup (PS \setminus \overline{AS})$. 用这个新的方程组序列 PS_i 替换 PS, 返回第 1 步.

Step 4 计算 AS 关于 g 的 **WR** 分解得到一系列新的正常升列 AS_i.

Step 5 计算 g 关于每一组正常升列 AS_i 的结式 g_i. 如果 $g_i \neq 0$, 那么构造新的方程组 $PS_{i0} = AS_i \bigcup \{g_i\} \bigcup (PS \setminus AS \setminus \{g\})$; 否则 $PS_{i0} = AS_i \bigcup (PS \setminus AS \setminus \{g\})$, $PS_{ij} = AS_i \bigcup (PS \setminus AS) \bigcup C_{ij}$, 此处 C_{ij} 表示升列 AS_i 的第 j 个方程关于其导元的所有系数之集.

Step 6 用新的方程组序列 PS_{ij} 替换 PS, 返回第 1 步.

Step 7 如果 $Aset = \{\ \}$, 输出空集停止. 否则对 $Aset$ 中每个方程组关于原方程组的每个方程做 **WR** 相对分解, 去掉多余零点后, 输出新的三角型方程组集合 $Aset$ 停止. 这里值得一提的是, 考虑到变量的序对整序的复杂度影响很大, 在上面 **WRSOLVE** 算法中, 并不要求预先人为指定变量序, 所有的序都是由算法自动产生, 这点与其他已有的算法不同.

[例 1] 一个由摩勒定理的假设所产生的方程组

著名的摩勒 (Morley) 定理是关于一个三角形的诸三等分角线的某三个交点构成一个等边三角形的定理. 周咸青[8] 将这个几何命题的假设 (通过引进适当坐标系) 加以代数化, 得到一个依赖于约束变量 $x_1, \cdots, x_6$ 和参数 u_1, u_2, u_3 的方程组如下:

$$
\begin{aligned}
h_1 = &(u_3^3 + (-3u_2^2 + 6u_1u_2 - 3u_1^2)u_3)x_2 + \\
&((-3u_2 + 3u_1)u_3^2 + u_2^3 - 3u_1u_2^2 + 3u_1^2u_2 - u_1^3)x_1 \\
&-u_1u_3^3 + (3u_1u_2^2 - 6u_1^2u_2 + 3u_1^3)u_3, \\
h_2 = &(u_3^3 - 3u_2^2u_3)x_2 + (-3u_2u_3^2 + u_2^3)x_1, \\
h_3 = &(u_1u_3x_2 - u_1u_2x_1)x_4 + (u_1u_2x_2 + u_1u_3x_1)x_3, \\
h_4 = &(3x_1x_2^2 - 2u_1x_1x_2 - x_1^3)x_4^3 + ((-3x_2^3 + 3u_1x_2^2 + \\
&9x_1^2x_2 - 3u_1x_1^2)x_3 - 6x_1x_2^3 + 3u_1x_1x_2^2 - 6x_1^3x_2 \\
&+3u_1x_1^3)x_4^2 + ((-9x_1x_2^2 + 6u_1x_1x_2 + 3x_1^3)x_3^2 + \\
&(6x_2^4 - 6u_1x_2^3 - 6u_1x_1^2x_2 - 6x_1^4)x_3 + 3x_1x_2^4 + \\
&6x_1^3x_2^2 + 3x_1^5)x_4 + (x_2^3 - u_1x_2^2 - 3x_1^2x_2 + u_1x_1^2)x_3^3 \\
&+(6x_1x_2^3 - 3u_1x_1x_2^2 + 6x_1^3x_2 - 3u_1x_1^3)x_3^2 + \\
&(-3x_2^5 + 3u_1x_2^4 - 6x_1^2x_2^3 + 6u_1x_1^2x_2^2 - 3x_1^4x_2 + \\
&3u_1x_1^4)x_3 - u_1x_1x_2^4 - 2u_1x_1^3x_2^2 - u_1x_1^5, \\
h_5 = &(u_1u_3x_2 + (-u_1u_2 + u_1^2)x_1 - u_1^2u_3)x_6 + \\
&((u_1u_2 - u_1^2)x_2 + u_1u_3x_1 - u_1^2u_2 + u_1^3)x_5 - \\
&u_1^2u_3x_2 + (u_1^2u_2 - u_1^3)x_1 + u_1^3u_3, \\
h_6 = &((2x_1x_2 - u_1x_1)x_4 + (-x_2^2 + u_1x_2 + x_1^2)x_3 - x_1x_2^2 - \\
&x_1^3)x_6 + ((-x_2^2 + u_1x_2 + x_1^2)x_4 + (-2x_1x_2 + u_1x_1)x_3 \\
&+x_2^3 - u_1x_2^2 + x_1^2x_2 - u_1x_1^2)x_5 + (-x_1x_2^2 - x_1^3)x_4 + \\
&(x_2^3 - u_1x_2^2 + x_1^2x_2u_1x_1^2)x_3 + u_1x_1x_2^2 + u_1x_1^3.
\end{aligned}
$$

如果使用附录 C 中的 WRSOLVE 程序来对上述方程组整序，只需键入

```
wrsolve([h1,h2,h3,h4,h5,h6],[x1,x2,x3,x4,x5,x6]);
```

机器将会自动进行整序，最后自动产生一个升列，其零点包括了原方程组的所有零点．整个过程在 PC 586/120 微机上费时仅 3 秒．

§ 33 麦 考 莱 商

给定 k 个 $k-1$ 变元的多项式组

$$
PS:\begin{cases}
p_1(x_1,x_2,\ldots,x_{k-1}),\\
p_2(x_1,x_2,\ldots,x_{k-1}),\\
\cdots\cdots\cdots\cdots\cdots\cdots\\
p_k(x_1,x_2,\ldots,x_{k-1}).
\end{cases}
$$

设它们的次数分别是 $n_1, n_2, \ldots, n_k$. 取 x_k 为一新变元, 令

$$PS: \begin{cases} f_1 = x_k^{n_1} p_1(\frac{x_1}{x_k}, \frac{x_2}{x_k}, \ldots, \frac{x_{k-1}}{x_k}), \\ f_2 = x_k^{n_2} p_2(\frac{x_1}{x_k}, \frac{x_2}{x_k}, \ldots, \frac{x_{k-1}}{x_k}), \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ f_k = x_k^{n_k} p_k(\frac{x_1}{x_k}, \frac{x_2}{x_k}, \ldots, \frac{x_{k-1}}{x_k}). \end{cases}$$

则多项式组 $f_1, f_2, \ldots, f_k$ 是关于变元 $x_1, x_2, \ldots, x_k$ 的齐次多项式, 且次数分别是 $n_1, n_2, \ldots, n_k$.

早在 1902 年, 对于 k 个一般的 k 变元齐次多项式组 ($k \geq 2$), 麦考莱发现了一个形式非常简明的表达式. 按照麦考莱的方法, 结式可以表达为两个行列式的商, 称之为麦考莱商, 其分母是分子的一个子式. 当所有这些多项式都是线性式或者多项式只有两个时, 这个分母是域的一个单位, 麦考莱商可以看作为线性情形的克莱姆法和两个多项式情形的西尔维斯特结式的自然推广. 当所有多项式的次数都相等时, 麦考莱商的分子和分母所对应行列式的阶数较之一般情形要小[28].

本节描述这两个麦考莱算法, 其证明及其它有关论断请参阅 [27],[28]. 我们把麦考莱商的分子和分母所对应的矩阵分别记为 M 和 N. 至于具体指定多项式组的情形, 我们在下一节将通过一个例子给出一种有用的处理方法.

§ 33.1 一般次数情形

设 $f_1, f_2, \ldots, f_k$ 是关于变元 $x_1, x_2, \ldots, x_k$ 的齐次多项式, 且次数分别是 $n_1, n_2, \ldots, n_k$. 令

$$m = 1 + \sum_{i=1}^{k} (n_i - 1),$$
$$X = \{x_1^{i_1} x_2^{i_2} \cdots x_k^{i_k} \mid i_1 + i_2 + \cdots + i_k = m\}.$$

换句话说, X 是所有关于变元 $x_1, x_2, \ldots, x_k$ 的 m 次齐次幂积的集合.

给定有序组 $(f_1,\cdots,f_k;\ x_1,\cdots,x_k)$, 令

$$\begin{aligned}
&T_0 = X,\\
&S_1 = \{x \in T_0 |\ x_1^{n_1} \text{整除} x\},\\
&X_1 = \{x |\ x_1^{n_1} x \in S_1\},\\
&T_1 = T_0 \setminus S_1,\\
&S_2 = \{x \in T_1 |\ x_2^{n_2} \text{整除} x\},\\
&X_2 = \{x |\ x_2^{n_2} x \in S_2\},\\
&T_2 = T_1 \setminus S_2,\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&S_k = \{x \in T_{k-1} |\ x_k^{n_k} \text{整除} x\}\\
&X_k = \{x |\ x_k^{n_k} x \in S_k\},\\
&T_k = T_{k-1} \setminus S_k\ =\ \emptyset.
\end{aligned}$$

这样, 就把集合 X 划分为

$$X = x_1^{n_1} X_1 \bigcup x_2^{n_2} X_2 \bigcup \cdots \bigcup x_k^{n_k} X_k.$$

然后, 用 X_i 中的每个幂积 e_j(其次数为 $m-n_i$) 乘以 f_i $(i=1,2,\ldots,k)$, 最后得到 $\begin{pmatrix} m+k-1 \\ k-1 \end{pmatrix}$ 个 m 次的齐次多项式. 我们把这组多项式称为 $(f_1,f_2,\ldots,f_k)$ 的 **麦考莱导出多项式组**. 把 X 中的所有幂积按字典序 (或全幂序) 由大到小写为

$$e_n,\ \cdots,\ e_2,\ e_1,$$

则麦考莱导出多项式组可写为标准形式:

$$M \cdot \begin{pmatrix} e_n \\ \vdots \\ e_2 \\ e_1 \end{pmatrix}$$

其系数矩阵 M 称为多项式组 $(f_1,f_2,\ldots,f_k)$ 关于变元 $(x_1,x_2,\cdots,x_k)$ 的 **麦考莱矩阵**. 我们分别用 $e_jf_i(e_j \in X_i)$ 和 e_i 作为这个矩阵的

行指标和列指标, 即

$$M = \begin{pmatrix} & & & \\ & & & \\ & & & \end{pmatrix} \begin{matrix} \}X_1 f_1 \\ \}X_2 f_2 \\ \vdots \\ \}X_k f_k \end{matrix}$$

（列指标：$e_n \quad \cdots \quad e_2 \quad e_1$）

令

$$F_1 = \{x \in X_1 | \text{ 在 } x_2^{n_2}, \cdots, x_k^{n_k} \text{ 中至少存在一个整除 } x\},$$
$$F_2 = \{x \in X_2 | \text{ 在 } x_3^{n_3}, \cdots, x_k^{n_k} \text{ 中至少存在一个整除 } x\},$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$F_{k-1} = \{x \in X_{k-1} | \; x_k^{n_k} \text{ 整除 } x\}.$$

我们把麦考莱矩阵中, 对应于行指标为 $F_1 f_1, \cdots, F_{k-1} f_{k-1}$, 而列指标为 $x_1^{n_1} F_1, \cdots, x_{k-1}^{n_k} F_{k-1}$ 的子矩阵称为 **麦考莱子矩阵**, 我们将其记之为 N.

有序组 $(f_1, \cdots, f_k; \; x_1, \cdots, x_k)$ 的麦考莱商就是

$$\frac{\det(M)}{\det(N)}.$$

§ 33.2 相等次数情形

设 $f_1, f_2, \ldots, f_k$ 是关于变元 $x_1, x_2, \ldots, x_k$ 的齐次多项式, 且它们的次数都是 n. 给定 $(f_1, \cdots, f_k; \; x_1, \cdots, x_k)$, 令

$$d = (k-1)(n-1),$$
$$X_1^* = \{x_1^{i_1} \cdots x_k^{i_k} | i_1 + \cdots + i_k = d,$$
$$i_1 + \cdots + i_j \le j(n-1), 1 \le j \le k-1\}.$$

集合 X_1^* 中共有 $\frac{n}{k-1}\binom{kn-1}{k-2}$ 个幂积.

令

$$
\begin{aligned}
X_2^* &= X_1^*,\\
X_3^* &= \{x_1^{i_1}\cdots x_k^{i_k}\in X_2^* | i_2 < n\},\\
X_4^* &= \{x_1^{i_1}\cdots x_k^{i_k}\in X_3^* | i_3 < n\},\\
&\cdots\cdots\cdots\cdots\\
X_k^* &= \{x_1^{i_1}\cdots x_k^{i_k}\in X_{k-1}^* | i_{k-1} < n\},\\
X^* &= x_1^n X_1^* \bigcup x_2^n X_2^* \bigcup\cdots\bigcup x_k^n X_k^*.
\end{aligned}
$$

用 X_i^* 中的每个幂积 e_j 乘以 $f_i(i=1,2,\ldots,k)$,得到一组多项式,我们把这组多项式称为 $(f_1,\cdots,f_k;\ x_1,\cdots,x_k)$ 的 **麦考莱导出多项式组**. 把 X^* 中的所有幂积按字典序 (或全幂序) 由大到小写为

$$e_n,\ \cdots,\ e_2,\ e_1,$$

则麦考莱导出多项式组可写为

$$M^*\cdot\begin{pmatrix} e_n\\ \vdots\\ e_2\\ e_1\end{pmatrix}.$$

其系数矩阵 M^* 相应称为 **麦考莱矩阵**. 我们分别用 $e_jf_i(e_j\in X_i)$ 和 e_i 作为这个矩阵的行指标和列指标.

令

$$
\begin{aligned}
F_2 &= \{x\in X_1^* |\ \text{在}\ x_2^n,\cdots,x_k^n\text{中至少存在一个整除}\ x\},\\
F_3 &= \{x\in X_2^* |\ \text{在}\ x_3^n,\cdots,x_k^n\text{中至少存在一个整除}\ x\},\\
&\cdots\cdots\cdots\cdots\\
F_{k-1} &= \{x\in X_{k-1}^* |\ x_k^n\ \text{整除}\ x\},\\
F_1 &= F_2.
\end{aligned}
$$

我们把麦考莱矩阵中, 对应于行指标为 $F_1f_1,\ \cdots,\ F_{k-1}f_{k-1}$, 而列指标为 $x_1^nF_1,\ \cdots,\ x_{k-1}^nF_{k-1}$ 的子矩阵称为 **麦考莱子矩阵**, 记之为 N^*.

有序组 $(f_1,\cdots,f_k;\ x_1,\cdots,x_k)$ 的麦考莱商就是

$$\frac{\det(M^*)}{\det(N^*)}.$$

注记

(1) $\det(M^*)$ 和 $\det(N^*)$ 分别是 $\det(M)$ 和 $\det(N)$ 的因子.

(2) $X_k^* = X_k$.[28] 让 $\#(S)$ 表示集合 S 的势; 由前述构造知, $\#(X_1) - \#(X_1^*)$ 等于总次数为 d 且关于 x_1 次数至少为 n 的所有幂积的个数, 因此

$$\#(X_1) - \#(X_1^*) = \binom{d-n+k-1}{k-1} = \binom{n(k-2)}{k-1}.$$

(3) 设 σ 和 τ 是 $1,2,\cdots,k$ 的两个轮换, $M_{\sigma,\tau}$ 和 $N_{\sigma,\tau}$ 是运用前述算法于

$$(f_{\sigma(1)},\cdots,f_{\sigma(k)};\ x_{\tau(1)},\cdots,x_{\tau(k)})$$

而得到的麦考莱矩阵和麦考莱子矩阵. 这种组合共有 $(k!)^2$ 组, 一般来说它们是不同的, 但所有的麦考莱商是相等的 (除去因子 ± 1 之外).

§ 34 麦考莱商的例

本节通过几个例子来了解构造麦考莱商的具体过程, 并说明在实际计算时怎样解决分母为零的问题.

[例 1] 考虑如下由两个二次式和一个一次式组成的系统:

$$\begin{aligned} f_1 &= a_1x_1^2 + a_2x_1x_2 + a_3x_1x_3 + a_4x_2^2 + a_5x_2x_3 + a_6x_3^2, \\ f_2 &= b_1x_1^2 + b_2x_1x_2 + b_3x_1x_3 + b_4x_2^2 + b_5x_2x_3 + b_6x_3^2, \\ f_3 &= c_1x_1 + c_2x_2 + c_3x_3. \end{aligned}$$

此时 $k=3,\ n_1=2,\ n_2=2,\ n_3=1$.

考虑有序组 $(f_1, f_2, f_3;\ x_1, f_2, f_3)$，我们有：

$$\begin{aligned}
&m = 3,\\
&X = \{x_1^3, x_1^2x_2, x_1^2x_3, x_1x_2^2, x_1x_2x_3, x_1x_3^2, x_2^3, x_2^2x_3, x_2x_3^2, x_3^3\},\\
&X_1 = \{x_1, x_2, x_3\},\\
&X_2 = \{x_1, x_2, x_3\},\\
&X_3 = \{x_1x_2, x_1x_3, x_2x_3, x_3^2\}.
\end{aligned}$$

所以，有序组 $(f_1, f_2, f_3;\ x_1, f_2, f_3)$ 的麦考莱矩阵

$$M = \begin{array}{c} \begin{array}{cccccccccc} x_1^3 & x_1^2x_2 & x_1^2x_3 & x_1x_2^2 & x_1x_2x_3 & x_1x_3^2 & x_2^3 & x_2^2x_3 & x_2x_3^2 & x_3^3 \end{array} \\ \left(\begin{array}{cccccccccc}
a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & 0 & 0 & 0 & 0\\
0 & a_1 & 0 & a_2 & a_3 & 0 & a_4 & a_5 & a_6 & 0\\
0 & 0 & a_1 & 0 & a_2 & a_3 & 0 & a_4 & a_5 & a_6\\
b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & 0 & 0 & 0 & 0\\
0 & b_1 & 0 & b_2 & b_3 & 0 & b_4 & b_5 & b_6 & 0\\
0 & 0 & b_1 & 0 & b_2 & b_3 & 0 & b_4 & b_5 & b_6\\
0 & c_1 & 0 & c_2 & c_3 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & c_1 & 0 & c_2 & c_3 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & c_1 & 0 & 0 & c_2 & c_3 & 0\\
0 & 0 & 0 & 0 & 0 & c_1 & 0 & 0 & c_2 & c_3
\end{array}\right)\end{array}
\begin{array}{l} \\ x_1f_1\\ x_2f_1\\ x_3f_1\\ x_1f_2\\ x_2f_2\\ x_3f_2\\ x_1x_2f_3\\ x_1x_3f_3\\ x_2x_3f_3\\ x_3^2f_3 \end{array}$$

又 $F_1 = \{x_3\}$，$F_2 = \{x_3\}$，所以相应的麦考莱子矩阵

$$N = \begin{array}{c} \begin{array}{cc} x_1^2x_3 & x_2^3 \end{array} \\ \left(\begin{array}{cc} a_1 & a_4\\ b_1 & b_4 \end{array}\right) \end{array} \begin{array}{l} \\ x_3f_1\\ x_3f_2 \end{array}$$

为了简化计算，在用麦考莱商 $|M|/|N|$ 来求具体指定多项式组的结式时，常常要把一般的文字系数换为具体指定的系数，先化简有关行列式再进行展开和相除，而不是相反．可是这样做有时会遇到 0/0 型，即麦考莱商的分母 $|N| = 0$ 的情形．

克服这一困难的一个办法是试试所有 $(k!)^2$ 个麦考莱商，这样做有时是可行的．可是，也有可能出现所有的这 $(k!)^2$ 个麦考莱商都是 0/0 型．

[例 2] 考虑如下类型的多项式组

$$
\begin{aligned}
p_1 &= f(r,s,t) - xq(r,s,t),\\
p_2 &= g(r,s,t) - yq(r,s,t),\\
p_3 &= h(r,s,t) - zq(r,s,t).
\end{aligned}
$$

其中 x, y, z 为参数.

当

$$
f(r,s,t) = rs + t^2, \quad g(r,s,t) = rt + s^2,
$$
$$
h(r,s,t) = st, \qquad q(r,s,t) = r^2
$$

时, 所有 36 种组合的麦考莱商之分母 $|N|$ 如下表所示:

$\lvert N\rvert$	(r,s,t)	(r,t,s)	(s,r,t)	(s,t,r)	(t,r,s)	(r,t,s)
(p_1,p_2,p_3)	x^2	$-xy$	0	0	$-y$	1
(p_1,p_3,p_2)	0	$-xz$	0	0	$-z$	0
(p_2,p_1,p_3)	$-xy$	y^2	x	-1	0	0
(p_2,p_3,p_1)	$-yz$	0	z	0	0	0
(p_3,p_1,p_2)	0	z^2	0	0	0	0
(p_3,p_2,p_1)	z^2	0	0	0	0	0

而当

$$
\begin{aligned}
f(r,s,t) &= r^2 + s^2 + t^2 + rs,\\
g(r,s,t) &= r^2 + s^2 + t^2 + rt,\\
h(r,s,t) &= r^2 + s^2 + t^2 + st,\\
q(r,s,t) &= r^2 + s^2 + t^2.
\end{aligned}
$$

时, 所有 36 个麦考莱商之分母都为 0, 尽管结式本身并不为 0.

在麦考莱商之分母为 0 时, 也可以用所谓"扰动方法"来计算结式.

考虑上述例 2 的后一种情形, 它的一般型由三个二次型组成:

$$
\begin{aligned}
p_1 &= a_1r^2 + a_2rs + a_3rt + a_4s^2 + a_5st + a_6t^2,\\
p_2 &= b_1r^2 + b_2rs + b_3rt + b_4s^2 + b_5st + b_6t^2,\\
p_3 &= c_1r^2 + c_2rs + c_3rt + c_4s^2 + c_5st + c_6t^2.
\end{aligned}
$$

其中

$$
\begin{aligned}
&a_1 = a_4 = a_6 = 1 - x,\\
&b_1 = b_4 = b_6 = 1 - y,\\
&c_1 = c_4 = c_6 = 1 - z,\\
&a_2 = b_3 = c_5 = 1,\\
&a_3 = a_5 = b_2 = b_5 = c_2 = c_3 = 0.
\end{aligned}
$$

此时麦考莱商的分母是

$$
|N| = \begin{vmatrix} a_1 & 0 & a_6 \\ 0 & a_1 & a_4 \\ 0 & b_1 & b_4 \end{vmatrix} = \begin{vmatrix} 1-x & 0 & 1-x \\ 0 & 1-x & 1-x \\ 0 & 1-y & 1-y \end{vmatrix} = 0.
$$

让我们"扰动"b_4 这一项, 以使得 $|N|$ 不为 0. 令 $b_4 = b$, 则

$$
|N| = (1-x)^2(b+y-1).
$$

麦考莱商的分子

$$
\begin{aligned}
|M| = \ & (1-x)^2(b+y-1)(9 - 4z^2y^2x + 14z^2xy - 18x - 9b\\
&+x^4 - 12y - 18z - 2z^3xy - 12z^2x + 8y^2x^2 + 15z^2\\
&+24zy - 22z^2y + 4y^2 - 8zy^2 + 8z^2y^2 + 10z^3y - 4z^3y^2\\
&+2z^3x - 2z^4y + z^4y^2 - 6z^3 + z^4 - 28zxy + 2z^2y^2x^2\\
&-4zy^2x^2 + 3z^2x^2 - 2yx^4 + 24xy - 4z^2yx^2 + 24zx\\
&+8zy^2x - 8y^2x + y^2x^4 + 15x^2 - 22yx^2 - 6x^3 - 12zx^2\\
&+14zx^2y - 2zyx^3 + 6by + 2zx^3 + 10yx^3 - 4y^2x^3\\
&-2z^3bx - 8zbyx^2 - 8bz^2xy + 8b^2x^2 + 8b^2z^2 - 14bxy\\
&-4z^2b^2x - 2bx^4 - 8b^2z - 27zbx + 21bx + 14z^2bx - 8b^2x\\
&+2z^4by + 15bz^2y - 8bz^3y + 8zb^2x + 21bz - 14bzy\\
&+4b^2 - 21bz^2 + b^2x^4 + 10bx^3 - 2z^4b - 2bzx^3 - 8byx^3\\
&+2z^2b^2x^2 + z^4b^2 + 10bz^3 - 4z^2x^2b - 4b^2x^3\\
&-4zb^2x^2 + 15byx^2 + 14zbx^2 + 4z^2x^2by - 4b^2z^3\\
&+2byx^4 - 21bx^2 + 16bzxy).
\end{aligned}
$$

做除法之后令 $b = 1 - y$ 即得结式:

$$
\begin{aligned}
&4 - 5(x+y-z) + 5(xy+xz+yz) + 2(x^2+y^2+z^2)\\
&-xyz - 2x(y^2+z^2) - 2y(x^2+z^2) - 2(x^2+y^2)\\
&+x^2y^2 + x^2z^2 + y^2z^2.
\end{aligned}
$$

§ 35 矩阵广义特征值方法

有许多方法可以计算两个单变元多项式的结式，我们在第 2 章已作了讨论. 其中有一个所谓友阵方法，它在多变元情形对应于一个有价值的方法，就是所谓矩阵广义特征值方法，本节对此作一简要介绍.

设 $A_0, A_1, \ldots, A_n$ 是 $m \times m$ 矩阵，λ 是一未定元，我们称形式

$$L(\lambda) = A_0\lambda^n + A_1\lambda^{n-1} + \cdots + A_n.$$

为 n 次矩阵多项式. 当 λ 在基域 K 中取值时，按矩阵运算这个多项式显然是一个矩阵. 当 A_0 为单位阵时，相应的矩阵多项式称作是首一的.

让我们考虑 A_0 为可逆阵的情形. 令

$$\begin{aligned} \overline{L}(\lambda) &= A_0^{-1}L(\lambda), \\ \overline{A}_i &= A_0^{-1}A_i. \end{aligned} \qquad 0 < i \le n$$

则 $\overline{L}(\lambda)$ 是首一多项式，这个多项式作为一个方阵，其行列式是如下方阵的特征多项式：

$$C = \begin{pmatrix} 0 & I_m & 0 & \cdots & 0 \\ 0 & 0 & I_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I_m \\ -\overline{A}_n & -\overline{A}_{n-1} & -\overline{A}_{n-2} & \cdots & -\overline{A}_1 \end{pmatrix}$$

其中 0 和 I_m 分别为 $m \times m$ 零矩阵和单位阵. 我们把这个方阵称之为矩阵多项式 $\overline{L}(\lambda)$ 的 **块友阵**. 换句话说，矩阵多项式的行列式恰好是其块友阵的特征多项式.

当 A_0 是奇异阵时，考虑矩阵多项式

$$C_L(\lambda) = P\lambda - Q.$$

其中

$$P = \begin{pmatrix} I_m & 0 & 0 & \cdots & 0 \\ 0 & I_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & A_1 \end{pmatrix},$$

$$Q = \begin{pmatrix} 0 & I_m & 0 & \cdots & 0 \\ 0 & 0 & I_m & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I_m \\ -A_n & -A_{n-1} & -A_{n-2} & \cdots & -A_1 \end{pmatrix}$$

可以证明

$$\det(C_L(\lambda)) = \det(L(\lambda)).$$

多项式 $\det(L(\lambda))$ 的根称为矩阵对 (P,Q) 的特征值. 当 P 为单位阵时, 矩阵对 (P,Q) 的特征值就是 Q 的特征值. 这样我们就把多项式 $\det(L(\lambda))$ 求根问题化成了求矩阵对 (P,Q) 的特征值问题, 即线性代数理论中所谓矩阵广义特征值问题.

当 A_0 奇异而 A_n 非奇异时, 通常的做法是作变换 $\lambda = \beta^{-1}$, 从而把问题归结为 A_0 非奇异的情形.

下面来看多项式方程组求解问题如何能够化作矩阵广义特征值问题.

令

$$PS: \begin{cases} p_1 = p_1(x_1, x_2, \ldots, x_k) = 0, \\ p_2 = p_2(x_1, x_2, \ldots, x_k) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ p_k = p_k(x_1, x_2, \ldots, x_k) = 0. \end{cases}$$

用结式方法 (如迪克逊结式、麦考莱结式等), 构造 $(p_1, p_2, \cdots, p_k)$ 关于变元 $(x_2, \cdots, x_k)$ 的结式矩阵 (即从这 k 个多项式中保留 x_1 而消去 $k-1$ 个变元 $x_2, \cdots, x_k$). 一般来说, 构造结式矩阵并不是件困难的事, 所以这一步是不成问题的. 设所构造的结式矩阵为 $M(x_1)$,

显然它的每一元素都是 x_1 的多项式. 为了求解原方程组, 关键的一步是求 x_1 的多项式 $\det(M(x_1))$ 的根, 可是有时展开这个行列式是异常困难的, 此时可以把 $M(x_1)$ 写成矩阵多项式:

$$M(x_1) = A_0x_1^n + A_1x_1^{n-1} + \cdots + A_n.$$

从前面的分析知, 我们可以把问题化成求矩阵对 (P, Q) 的特征值问题.

关于特征值方法的近期研究, 可参看 [17]、[23].

§ 36 伯恩斯坦定理

如所周知, 代数学基本定理是说: 每个 n 次复系数多项式

$$f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n,$$

(如果 $a_0 \neq 0$) 在复数域中有且仅有 n 个根. 从名字可以看出, 这个定理在代数学中曾处于很重要的位置. 它在多个变元情形的推广, 一般认为是贝佐定理: 次数分别是 $n_1, n_2, \cdots, n_k$ 的 k 个 k 变元多项式组

$$PS: \begin{cases} p_1 = p_1(x_1, x_2, \ldots, x_k), \\ p_2 = p_2(x_1, x_2, \ldots, x_k), \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ p_k = p_k(x_1, x_2, \ldots, x_k). \end{cases}$$

在复射影空间 $P^k(\mathbf{C})$ 中恰有 $n_1n_2\cdots n_k$ 个公共零点 (如果公共零点的个数有限); 这是代数几何学的开卷定理. 然而,在一般情况下, 人们更关心 PS 在仿射空间 $A^k(\mathbf{C})$ 中有多少个公共零点. 由于实际遇到的方程组具有各种各样特殊的形状, 各个方程中出现的幂积并不齐全, 其解的个数亦不尽相同, 我们希望不解出方程组而仅仅根据它的一些简单特征来估计解的个数; 伯恩斯坦 (D.N.Bernstein) 定理[6] 给出了这个数目的一个很好上界.

先引入几个概念.

如同前面几节，我们用 e_i 来表示 $x_1, x_2, \cdots, x_k$ 的幂积. 一个多项式 $f = f(x_1, x_2, \ldots, x_k)$ 总可以写为其中所出现的幂积的线性组合：

$$f = \sum_{(i_1,\cdots,i_k)} c_{(i_1,\cdots,i_k)} e_{(i_1,\cdots,i_k)}.$$

令

$$S(f) = \{(i_1, \cdots, i_k) | c_{(i_1,\cdots,i_k)} \neq 0\},$$

称之为多项式 f 的 **支撑集**. 用 $\mathbf{R}^k$ 表示 k 维实向量空间. 在几何上 $S(f)$ 表示 $\mathbf{R}^k$ 中的一个格点集. 显然, $S(f) \subset \mathbf{Z}^k$. $S(f)$ 在 $\mathbf{R}^k$ 中的凸包叫做多项式 f 的 **牛顿** (I.Newton) **多胞形**, 记为 $N(f)$.

当我们说"多胞形"的时候, 总是意味着它是 $\mathbf{R}^k$ 中有限点集的凸包. 设 P、Q 是 $\mathbf{R}^k$ 中的两个多胞形, 记

$$P + Q = \{p + q | \ p \in P, \ q \in Q\},$$

我们称其为多胞形 P 和 Q 的 **闵可夫斯基** (H.Minkowski) **和**. 设 $\lambda \in \mathbf{R}$, 记

$$\lambda P = \{\lambda p | \ p \in P\}.$$

用 $\mathrm{Vol}(P)$ 表示多胞形 P(在 $\mathbf{R}^k$ 中) 的体积. 对于 $\mathbf{R}^k$ 中任何 k 个多胞形 $P_1, P_2, \cdots, P_k$, 表达式

$$\mathrm{Vol}(\lambda_1 P_1 + \lambda_2 P_2 + \cdots + \lambda_k P_k) \tag{36.1}$$

是关于 λ_1、$\lambda_2, \cdots, \lambda_k$ 的一个 k 次齐次多项式.

在多项式 (36.1) 中, 幂积 $\lambda_1 \lambda_2 \cdots \lambda_k$ 的系数叫做多胞形 P_1, $P_2, \cdots, P_k$ 的 **混合积**, 记为 $\mathrm{Vol}(P_1, P_2, \cdots, P_k)$.

更明显地, 我们有

$$\mathrm{Vol}(P_1, P_2, \cdots, P_k) = \frac{1}{k!} \sum_{j=1}^{k} (-1)^{k-j} \sum_{1 \le i_1 < \cdots < i_j \le k} \mathrm{Vol}(P_{i_1} + \cdots + P_{i_j}).$$

现在我们来给出伯恩斯坦定理 (表达与其原形式略有不同):

定理 1 k 个 k 变元多项式 $p_1, p_2, \cdots, p_k$ 在仿射空间 $A^k(\mathbf{C})$ 中公共无挠零点的个数不超过它们的牛顿多胞形的混合积

$$\mathrm{Vol}(N(p_1), N(p_2), \cdots, N(p_k)).$$

§ 37 多元结式的一些性质

本节介绍有关多变元结式的一些重要关系和结论. 我们把 §12 的方法推广到多个变元的情形, 推导过程是完全相仿的.

考虑构造有序组 $(f_1, \cdots, f_k;\ x_1, \cdots, x_k)$ 的麦考莱商的过程.

沿用 §12 的有关记号. 记

$$n = n_1 n_2 \cdots n_{k-1}, \qquad l = \begin{pmatrix} m+k-1 \\ k-1 \end{pmatrix}.$$

线性方程组 $(\lambda = f_k)$:

$$M(\lambda) \cdot \begin{pmatrix} e_n \\ e_{n-1} \\ \vdots \\ e_1 \end{pmatrix} = \begin{pmatrix} X_1 f_1 \\ X_2 f_2 \\ \vdots \\ X_k(f_k - \lambda) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

其中

$$M(\lambda) = \begin{array}{c} \begin{matrix} e_n & \cdots & e_2 & e_1 \end{matrix} \\ \left(\begin{matrix} \quad & & & \\ & & & \\ & & & \end{matrix} \right) \end{array} \begin{matrix} \}X_1 f_1 \\ \}X_2 f_2 \\ \vdots \\ \}X_k(f_k - \lambda) \end{matrix}$$

这个方程组由 l 个方程组成, 它的前 $l-n$ 个方程仅对于方程组

$$\{f_1 = 0, \quad \cdots, \quad f_{k-1} = 0\}$$

(不妨令 $x_k = 1$) 的所有 n 组解 $\alpha_1, \alpha_2, \ldots, \alpha_n$ 成立 (根据贝佐定理, 一般情形恰好是 n 组解), 而后 n 个方程是恒等式. 因此, 这个线性方程组当且仅当 $(x_1, \cdots, x_{k-1})$ 为方程组

$$\{f_1 = 0, \quad \cdots, \quad f_{k-1} = 0\}$$

的解时有解；并且由于有 $e_1 = x_k^m = 1$, 这些解自然都是非平凡解, 这意味着其系数矩阵 $M(\lambda)$ 是奇异阵, 即 $M(\lambda)$ 的行列式 (展开为 λ 的多项式) 等于 0:

$$\det(M(\lambda)) = c_0\lambda^n + c_1\lambda_{n-1} + \cdots + c_n = 0.$$

这就是说, $\det(M(\lambda))$ 作为 λ 的 n 次多项式, 它的 n 个根为

$$f_k(\alpha_1),\ f_k(\alpha_2),\ \ldots,\ f_k(\alpha_n).$$

我们把在多项式 f_i 中令 $x_k = 0$ 而得到的关于变元 $x_1, \cdots, x_{k-1}$ 的齐次多项式记作 $\tilde{f}_i(i = 1, \cdots, k-1)$.

容易知道, 在 $\det(M)$ 的上述展开式中,

$$c_n = |M|, \quad c_0 = (-1)^n |N| (\mathrm{res}(\tilde{f}_1, \cdots, \tilde{f}_{k-1}))^{n_k}.$$

由根与系数的关系, 得:

$$\prod_{i=1}^{n} f_k(\alpha_i) = (-1)^n \frac{c_n}{c_0}.$$

这样, 就得到一个重要关系 (证明细节参见 [18]):

$$\mathrm{res}(f_1, \cdots, f_k) = (\mathrm{res}(\tilde{f}_1, \cdots, \tilde{f}_{k-1}))^{n_k} \prod_{i=1}^{n} f_k(\alpha_i). \tag{37.1}$$

设 λ 是一个参变元, $f_1, \cdots, f_k$ 与 g 是关于变元 $x_1, \cdots, x_k$ 的多项式, 结式 $\mathrm{res}(f_1, \cdots, f_k, g + \lambda)$ 称为 $f_1, \cdots, f_k$ 与 g 的 λ 结式. 它是关于 λ 的一个多项式. 我们有:

定理 1 记号如上. 如果 $\mathrm{res}(\tilde{f}_1, \cdots, \tilde{f}_{k-1}) \neq 0$, 那么在方程组 $\{f_1 = 0, \cdots, f_k = 0\}$ 的所有解中恰好有 k 个使得 $g = 0$ 的充要条件是结式 $\mathrm{res}(f_1, \cdots, f_k, g + \lambda)$ 关于 λ 的最低次数为 k.

第 5 章

机器证明的例证法

计算机可以帮助人们进行推理和计算，这是一个明显的事实．同时，这也是一个强有力的思想，人们需要面向计算机来重新考察数学的各个方面．在这一广阔的背景里，涌现出了很多令人惊奇的事物．

这里介绍一个具有深刻哲学意味的发展：定理机器证明的数值方法．它可以看作是本书前面几章所发展的非线性代数方程组理论的一个应用和补充．

§ 38　概　述

读者不妨随意在纸上画一个三角形，并画这个三角形的三条角平分线，如果你作图足够精确，就会看到这三条角平分线相交于一点．这时你可以猜测这样一个结论：对于任意三角形，其三条角平分线相交于一点．多试几次可以增强你的信心．这种通过举例子做实验来发现数学定理和科学事实的方法，即是经验归纳法：从个别到一般，从具体到抽象．正是人类认识自然的基本方式之一．

但是发现归发现，证明却似乎是另一码事．从中学时代起，我们就开始接受传统的观念："举例子做实验完全不算是证明"．发现和证明的这种矛盾在人们心中凝成了一个千古谜结．哲学家穆勒 (J.S. Mill) 对此深感困惑，他问："为什么在有些场合，单个实例对完全归纳法是足够的，而在别的场合，大量并发的实例 (已知或推测没有任何单一的例外) 在确定一个全称命题方面却进展不大呢？"在科学研究中获得如此巨大成功的经验归纳法，究竟是建立在怎样的哲学基础之上的？

一个漂亮的回答来自机器证明领域. 洪加威[21]指出, 对于一类平面几何的定理, 只要按照一个简单的规则去举例子, 并且对这个具体的数值例子的计算 (或作图) 到一定的精确程度, 就完全可以用来精确地判定一个一般的几何命题. 这就是著名的 **例证法**. 例证法的原理适用于更广阔的范围, 只是洪加威所设想的算法其复杂度太高, 迄今未能在计算机上有效地实现.

随后出现了所谓 **数值并行法**[45],[46],[37]. 它在形式上更为接近归纳法, 其基本含义是: 具有某种一般性的大量实例可以证明一个一般的几何命题. 其思想是: 用数值计算代替符号计算以减少内存消耗; 用并行处理取代串行处理以缩短运行时间. 这是第一个具有实践意义的数值方法, 据此所编制的"L 类几何定理证明器"[44] 已经受了实践的检验, 可以在很短的时间内证明颇不平凡的定理. 作者还用它来检验人们提出的某些猜测, 从而发现了若干有趣的新定理.

大家知道, 过去已有的柯林斯 (G.E.Collins) 的柱形代数剖分 (CAD) 算法是一种用来验证不等式的例证法. 对此本章将不作详细介绍, 因为该算法的绝大部分工作量属于符号代数计算, 而本章主题是例证法中的数值方法.

近来又发展出一种效率较高、实际适用性很广的数值方法, 叫做 **单例实验法**,[22] 它第一次表明在一定意义下不准确的浮点运算可以用来作严格的推理, 具有实验的性质, 真正全面地实现了例证法的美妙设想.

数值方法的这些发展, 从一个方面向人们展示了经验归纳法的哲学基础: 面对一类自然现象, 当所期待的规律具有一定的数学形式时, 适当数量和适当精确度的实验足以确立之; 更多的数量和更高的精确度有利于确定更精细的规律.

本章介绍以上三种方法的基本原理以及后两者的具体实现.

§ 39 起 点

一个极为简单的事实, 等式

$$(x+1)(x-1)=x^2-1 \tag{39.1}$$

是一个恒等式. 只需把左端展开、合并, 便可证明.

但也可以用数值实验的方法证明. 取 $x=0$, 两端都是 -1; 取 $x=1$, 两端都是 0; 取 $x=2$, 两端都是 3. 这就证明了 (39.1) 是恒等式.

道理很简单, 如果它不是恒等式, 就是一个不高于二次的一元代数方程. 这种方程至多有两个根. 现在已有 $x=0,1,2$ 这三个根了, 那表明它不是方程而是恒等式.

推广到高次, 便是

命题 A 设 $f(x)$ 和 $g(x)$ 都是不超过 n 次的多项式. 如果有 $n+1$ 个不同的数 $a_0, a_1, \cdots, a_n$, 使得 $f(a_k)=g(a_k)\ (k=0,1,\cdots,n)$, 则等式

$$f(x)=g(x) \tag{39.2}$$

是恒等式.

通俗地说: 要问一个给定的不高于 n 次的一元代数等式是不是恒等式, 只要分别用 $n+1$ 个数值代替变元检验, 即可得出结论.

换句话, 举足够多的数值例子可以证明一元代数恒等式, 其基本依据是 n 次代数方程至多有 n 个根.

现在换一个角度来看等式 (39.1). 如果有人说, 只要取一个较大的 x 代入, 比如 $x=6$, 就足以证明它是恒等式! 这能行吗?

初看似乎令人吃惊, 细想也就不怪了. 在 (39.1) 式中, 左端展开后最多 4 项, 每项系数的绝对值至多为 1. 整理、合并之后, 系数的绝对值是不大于 5 的正整数或 0. 如果 (39.1) 不是恒等式, 把它整理之后, 应当是一个方程, 即

$$ax^2+bx+c=0. \tag{39.3}$$

这里 a,b,c 不全为 0, 都是绝对值不大于 5 的整数. 如果取 $x=6$ 时, 有

$$a\times 6^2+b\times 6+c=0, \tag{39.4}$$

可知 $a=b=c=0$.

若 $a=0$, 由 $6b+c=0$ 得 $6|b|=|c|$, 当 $b=0$ 时得 $c=0$. 当 $b\neq 0$ 时得 $|c|=6|b|\geq 6$, 这与 $|c|\leq 5$ 矛盾.

若 $a\neq 0$, 由 $36a+6b+c=0$ 得 $36|a|\leq 6|b|+|c|$, 即

$$36|a|\leq 6|b|+|c|\leq 35. \tag{39.5}$$

仍然矛盾.

这证明了 (39.1) 是恒等式.

这种办法也可以推广到高次. 更具体地有:

命题 B 设 $f(x)$ 和 $g(x)$ 都是不超过 n 次的多项式. 如果知道 $f(x)-g(x)$ 的标准展开式中系数绝对值最大者不大于 L, 非零系数绝对值最小者不小于 $S(>0)$. 设

$$|\hat{x}|=p\geq \frac{L}{S}+2,$$

则

$$S\leq |f(\hat{x})|-g(\hat{x})|\leq Sp^{n+1}. \tag{39.6}$$

证明 设

$$f(x)-g(x)=c_0x^k+c_1x^{k-1}+\cdots+c_k, \tag{39.7}$$

这里 $0\leq k\leq n$, 而 $c_0\neq 0$. 若 $k=0$, 显然有 (39.6) 成立.

以下设 $1\leq k\leq n$. 这时

$$\begin{aligned}
|f(\hat{x})|-g(\hat{x})| &\geq |c_0x^k|-|c_1x^{k-1}+c_2x^{k-2}+\cdots+c_k|\\
&\geq Sp^k-L(p^{k-1}+p^{k-2}+\cdots+p+1)\\
&\geq \tfrac{Sp^k}{p-1}\big(p-1-\tfrac{L}{S}\big(1-\tfrac{1}{p^k}\big)\big)\\
&\geq S\big(p-1-\tfrac{L}{S}\big)\geq S.
\end{aligned}$$

另一方面, 显然有 $|f(\hat{x})| - g(\hat{x})| \leq L\frac{p^{n+1}-1}{p-1} \geq Sp^{n+1}$.

不等式 (39.6) 表明, 对 $|\hat{x}| \geq \frac{L}{S} + 2$, $\hat{x}$ 不可能是方程 $f(x) - g(x) = 0$ 的根. 如果计算表明居然有 $f(\hat{x}) = g(\hat{x})$, 即可断言 $f(x) = g(x)$ 是恒等式 (至于不等式 (39.6) 的右端, 后面将用到).

通俗地说, 举一个足够复杂的数值例子, 即可证明一个代数恒等式. 其基本依据是: 代数方程的根的绝对值, 不超过其绝对值最大的系数与最高次项系数绝对值之比加 1(在命题 B 中, 采用了略强的条件, 是为了得到 (39.6) 中确定的正的下界, 这在后面将用到).

另一个特别有用的尝试是取 $x = \sqrt[3]{2}$. 如果它使得多项式

$$(x+1)(x-1) - x^2 + 1$$

取值为 0, 则可以断定该多项式是零多项式. 原因是多项式 $x^3 - 2$ 在有理数域上是不可约的, 所以它的根 $\sqrt[3]{2}$ 不可能是一个次数低于 3 的非零整系数多项式的根.

这种方法的高次推广是:

命题 C 设 $f(x)$ 和 $g(x)$ 都是不超过 n 次的多项式. 取 p 为任一素数, 如果 $f(p^{\frac{1}{n+1}}) = g(p^{\frac{1}{n+1}})$, 则等式

$$f(x) = g(x) \tag{39.8}$$

是恒等式.

§ 40 推 广

从前面所说的命题 A 、命题 B 及命题 C 起步, 发展到几何定理机器证明的并行法、例证法以及单例实验法, 需要更进一步的工作.

首先要做的是从一元推广到多元.

命题 A 的推广是:

定理 A 设 $f(x_1,x_2,\cdots,x_k)$ 是 $x_1,x_2,\cdots,x_k$ 的多项式，它关于 x_i 的次数不大于 n_i，对应于 $r=1,2,\cdots,k$，取数组 $u_{r,s}(s=0,1,\cdots,n_r)$，使得当 $s_1\neq s_2$ 时，有 $u_{r,s_1}\neq u_{r,s_2}$. 如果对任一组 $(s_1,s_2,\cdots,s_k)$，其中 $0\le s_i\le n_i$，有

$$f(u_{1,s_1},u_{2,s_2},\cdots,u_{k,s_k})=0, \tag{40.1}$$

则 $f(x_1,x_2,\cdots,x_k)$ 是恒为 0 的多项式.

证明 对 k 作数学归纳. 当 $k=1$ 时即 §39 中的命题 A. 设要证的命题对 $k=j$ 已真，往证它对 $k=j+1$ 也真. 这时把 $f(x_1,x_2,\cdots,x_{j+1})$ 写成

$$c_0x^n+c_1x^{n-1}+\cdots+c_n. \tag{40.2}$$

这里 $x=x_{j+1}$，$n=n_{j+1}$，而 $c_i=c_i(x_1,\cdots,x_j)$ 是关于 x_t 次数不大于 n_t 的多项式， $t=1,2,\cdots,j$.

取定 $\hat{x}_i=u_{i,s_i}(i=1,\cdots,j;\ 0\le s_i\le n_i)$，则 $f(\hat{x}_1,\cdots,\hat{x}_j,x)$ 是 x 的次数不超过 n 的多项式. 对于 x 的 $n+1$ 个不同的值 $u_{j+1,0},u_{j+1,1},\cdots,u_{j+1,n}$ 总有 $f(\hat{x}_1,\cdots,\hat{x}_j,u_{j+1,t})=0,t=0,1,\cdots,n$. 由命题 A 可知 $f(\hat{x}_1,\cdots,\hat{x}_j,x)$ 是零多项式，这表明 $c_i(\hat{x}_1,\cdots,\hat{x}_j)=0$. 又因为 $\hat{x}_i$ 可以是 $u_{i,0},u_{i,1},\cdots,u_{i,n_i}$ 中的任何一个，由归纳前提可知 $c_i(x_1,\cdots,x_j)$ 是零多项式. 由数学归纳法，命题得证.

为了便于理解，不妨看一下 $k=2$ 的特例. 要证明等式

$$(x+y)(x-y)=x^2-y^2$$

是恒等式. 注意到它关于 x,y 的次数都不大于 2，故可取 $x=0,1,2$ 和 $y=0,1,2$，分别组成变元 (x,y) 的 9 组值

$$(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2).$$

代入验算即可.

一般说来，在定理 A 的条件下，需要验算的变元数值共有 $(n_1+1)(n_2+1)\cdots(n_k+1)$ 组.

命题 B 的多元推广是：

定理 B 设 $f(x_1, x_2, \cdots, x_k)$ 是 $x_1, x_2, \cdots, x_k$ 的多项式, 它关于 x_i 的次数不大于 n_i, $1 \le i \le k$. 又设在它的标准展开式中系数绝对值最大者不大于 L, 非零系数绝对值最小者不小于 $S > 0$. 如果变元的一组值 $\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_k$ 满足

$$\begin{cases} |\hat{x}_1| = p_1 \ge \frac{L}{S} + 2, \\ |\hat{x}_2| = p_2 \ge p_1^{n_1+1} + 2, \\ |\hat{x}_3| = p_3 \ge p_2^{n_2+1} + 2, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ |\hat{x}_k| = p_k \ge p_k^{n_k+1} + 2, \end{cases}$$

则有

$$f(\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_k) \ge S > 0.$$

证明 对 k 作数学归纳. 当 $k = 1$ 时即命题 B. 设要证的命题对 $k-1$ 已真, 往证它对 k 也真. 记 $g(x_2, \cdots, x_k) = f(\hat{x}_1, x_2, \cdots, x_k)$, 则 g 是 $x_2, \cdots, x_k$ 这 $k-1$ 个变元的多项式, 它的系数具有形式

$$c(\hat{x}_1) = c_0\hat{x}_1^n + c_1\hat{x}_1^{n-1} + \cdots + c_n. \qquad (0 \le n \le n_1)$$

这些多项式 $c(x)$ 不都是零多项式, 系数 c_i 的绝对值最大者不超过 L, 非零者绝对值最小不小于 S. 由命题 B 可知, 当 $c(x)$ 非 0 多项式时

$$S \le c(\hat{x}_1) \le Sp_1^{n_1+1} = L_1.$$

而由题设条件知 $|\hat{x}_3| = p_2 \ge p_1^{n_1+1} + 2 = \frac{L}{S} + 2$. 由归纳前提可知

$$|g(\hat{x}_2, \cdots, \hat{x}_k)| = |f(\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_k)| \ge S.$$

证毕.

类似地, 命题 C 也可以推广到多元.

定理 C 设 $f(x_1, x_2, \cdots, x_k)$ 是 $x_1, x_2, \cdots, x_k$ 的多项式, 它关于 x_i 的次数不大于 n_i. 又 $p_1, p_2, \cdots, p_k$ 是 k 个互不相同的素数, 如果

$$f(p_1^{1/(n_1+1)}, p_2^{1/(n_2+1)}, \cdots, p_k^{1/(n_k+1)}) = 0,$$

则 $f(x_1,x_2,\cdots,x_k)$ 是恒为 0 的多项式.

这个定理的证明请参见 [51].

还有很多别的数值方法来判定多项式是否恒等, 但是真正有实际意义的并不多. 尽管定理 B 在实际运用中尚未取得效果, 但考虑到其历史价值, 本书在介绍其他两种方法的同时也对之作了简单介绍.

§ 41 几何命题的代数化

到现在为止, 我们只说明了通过验算一个或一些例子可以检验一个代数等式是不是恒等式, 这和大家感兴趣的几何定理的机器证明仍有相当大的差距!

如所周知, 通常初等几何中的命题, 如果在前提和结论中不牵涉不等式, 总可以用坐标法化成这样的问题: 已知有一些代数等式; 求证某一个代数等式成立. 当然这样的问题并不局限于初等几何命题, 而代数化的方法也是多种多样的. 我们先以熟知的西摩松 (R.Simson) 定理为例, 来说明如何用坐标法把一个几何命题化为一个代数命题.

西摩松定理 在 ΔABC 的外接圆上任取一点 D, 自 D 向 BC,CA,AB 引垂线, 垂足顺次为 E,F,G, 则 E,F,G 三点在一直线上.

这个定理涉及 7 个点: A,B,C,D,E,F,G. 设它们的笛卡尔坐标依次为:

$$A(x_1,y_1), B(x_2,y_2), C(x_3,y_3), D(x_4,y_4), E(x_5,y_5), F(x_6,y_6), G(x_7,y_7).$$

为了减少变元, 不妨取 ΔABC 的外接圆圆心为原点, 设圆半径为 1, 则可取 $D=(1,0)$; 再由 E 在 BC 上等已知条件, 可得:

$$\begin{cases} x_5=\lambda x_2+(1-\lambda)x_3, & y_5=\lambda y_2+(1-\lambda)y_3, \\ x_6=\mu x_3+(1-\mu)x_1, & y_6=\mu y_3+(1-\mu)y_1, \\ x_7=\rho x_1+(1-\rho)x_2, & y_7=\rho y_1+(1-\rho)y_2. \end{cases} \tag{41.1}$$

这样就可以用 λ, μ, ρ 三个参量代替 $x_5, x_6, x_7, y_5, y_6, y_7$ 这六个变量. 这时, 9 个变量 $x_1,x_2,x_3,y_1,y_2, y_3,\lambda,\mu,\rho$ 应满足 6 个等式:

$$
\begin{aligned}
p_1 &= x_1^2 + y_1^2 - 1, \\
p_2 &= x_2^2 + y_2^2 - 1, \\
p_3 &= x_3^2 + y_3^2 - 1, \\
p_4^* &= (1 - \lambda x_2 - (1-\lambda)x_3)(x_2 - x_3) - (\lambda y_2 + (1-\lambda)y_3)(y_2 - y_3), \\
p_5^* &= (1 - \mu x_3 - (1-\mu)x_1)(x_3 - x_1) - (\mu y_3 + (1-\mu)y_1)(y_3 - y_1), \\
p_4^* &= (1 - \rho x_1 - (1-\rho)x_2)(x_1 - x_2) - (\rho y_1 + (1-\rho)y_2)(y_1 - y_2).
\end{aligned}
$$

这里 p_1 表示 A 在单位圆上, p_4^* 表示 $DE \perp BC$ 等等. 要证的结论是 E, F, G 共直线, 即

$$
g = (x_5 - x_6)(y_5 - y_7) - (x_5 - x_7)(y_5 - y_6).
$$

利用 p_1,p_2,p_3 可把 p_4^*,p_5^*,p_6^* 变为:

$$
\begin{aligned}
p_4 &= (2\lambda - 1)(x_2x_3 + y_2y_3 - 1) + (x_2 - x_3) = 0, \\
p_5 &= (2\mu - 1)(x_3x_1 + y_3y_1 - 1) + (x_3 - x_1) = 0, \\
p_6 &= (2\rho - 1)(x_1x_2 + y_1y_2 - 1) + (x_1 - x_2) = 0.
\end{aligned}
$$

而在结论 g 中, 则可利用 (41.1) 消去 $x_5,y_5,\cdots$. 整理之, 可得:

$$
\begin{aligned}
g = &(\lambda\rho + \lambda\mu + \mu\rho - \lambda - \mu - \rho + 1) \\
&(x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)) = 0.
\end{aligned}
$$

这里 g 是一个含有 9 个变元的多项式. 但它的 9 个变元之间有 p_1, p_2, p_3, p_4, p_5, p_6 这 6 个方程联系着, 所以不能用 §40 所说的验算一个或若干个例子的方法判定它是不是恒为 0. 一般说来, 利用这 6 个方程可以从 g 中消去 6 个变元, 剩下 3 个自由变元. 从几何上看, 不妨取 x_1,x_2,x_3 为自由变元. 如果能够消去另外 6 个非自由变元而得到一个仅含 x_1,x_2,x_3 的多项式 $\Phi(x_1,x_2,x_3)$, 就可以从 $\Phi(x_1,x_2,x_3)$ 用 x_1,x_2,x_3 的数值来检验了.

§ 42 构造性几何命题

几何命题的代数化是不是真的像初看上去那样显得繁琐而困难呢？不是的！事实上，对于相当大一类初等几何命题，其代数表示可以取非常简单的形式．这类几何命题叫做（等式型）构造性几何命题，其中出现的点是由少数几条构造规则（比如尺规作图规则）一个点一个点地作出来的．§41 中所述西摩松定理就是这样的一个例子．本书主要讨论这类命题（虽然原理并不局限于此）．

我们先考虑由下面三部分组成的初等平面几何命题：

1. 选取自由点．
   Select($P_1, P_2, \cdots, P_s$)，在平面上任取 s 个点：$P_1, P_2, \cdots, P_s$．

2. 作交点．

   - P =Point1(P_1, P_2, P_3, P_4)，作直线 P_1P_2 与直线 P_3P_4 的交点．
   - P', P''=Point2(P_1, P_2, P_3, P_4)，作直线 P_1P_2 与圆 $C(P_3, P_4)$ 的交点，此处 $C(P_3, P_4)$ 表示以 P_3 为圆心而通过 P_4 的圆．
   - P', P''=Point3(P_1, P_2, P_3, P_4)，作圆 $C(P_1, P_2)$ 与圆 $C(P_3, P_4)$ 的交点．

3. 结论．
   一个形如 $P' = P''$ 的结论，此处 P' 和 P'' 是由上面两部分定义的两个点．

以上描述了用直尺和圆规所能构造的所有等式型几何命题．当然，为了实际构图的方便，可以选取更多的复合作图语句（比如在“单例实验法”的软件中设计有 31 条作图语句）．

下面是一个范例．

(西摩松定理):

circle(O)	(以 O 为圆心作圆);
circularpoint(D, A, B, C)	(在圆上取点 A, B, C, D);
foot(D, B, C, E)	(过 D 作 BC 的垂线, 垂足为 E);
foot(D, C, A, F)	(过 D 作 CA 的垂线, 垂足为 F);
foot(D, A, B, G)	(过 D 作 AB 的垂线, 垂足为 G);
collinear(E, F, G)	(结论是 E, F, G 在一直线上).

对于构造性几何命题来说, 其假设部分的代数表示容易化为一个三角型方程组 (这正对应于代数上的构造形式):

$$TS: \begin{cases} f_1(u_1, \cdots, u_r; x_1) = 0, \\ f_2(u_1, \cdots, u_r; x_1, x_2) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ f_s(u_1, \cdots, u_r; x_1, x_2, \cdots, x_s) = 0 \end{cases}$$

(且左边的多项式列组成一正常升列), 其中 $u_1, \cdots, u_r$ 对应于自由变元, 它们完全确定了几何构形; 而结论部分则对应于一个 (或数个) 多项式方程:

$$g(u_1, \cdots, u_r; x_1, \cdots, x_s) = 0.$$

在所对应的三角型方程组中, 如果每个 f_i 关于其主变元 x_i 的次数都是 1, 即

$$f_1 = a_1(u_1, \cdots, u_r)x_1 - b_1(u_1, \cdots, u_r),$$

$$f_i = a_i(u_1, \cdots, u_r; x_1, \cdots, x_{i-1})x_i - b_i(u_1, \cdots, u_r; x_1, \cdots, x_{i-1})$$

$(i = 2, \cdots, s)$, 那么这个子类叫做“L 类”(“线性类”). 在初等平面几何中, 绝大多数常见命题都可以叙述为“L 类”命题. 在尺规作图类所对应的三角型方程组中, 每个 $f_i(u_1, \cdots, u_r; x_1, \cdots, x_i)$ 关于其主变元 x_i 的次数是 2 或 1.

L 类构造性几何命题可以用几何的方式来描述:

1. 选取自由点.

   Select($P_1, P_2, \cdots, P_s$), 在平面上任取 s 个点: $P_1, P_2, \cdots, P_s$.

2. 作交点.

   - P =Point1(P_1, P_2, P_3, P_4), 作直线 P_1P_2 与直线 P_3P_4 的交点.
   - P =Point2(P_1, P_2, P_3, P_4), 已知直线 P_1P_2 与圆 $C(P_3, P_4)$ 的一个交点, 作另一个交点 P, 此处 $C(P_3, P_4)$ 表示以 P_3 为圆心而通过 P_4 的圆.
   - P=Point3(P_1, P_2, P_3, P_4), 已知圆 $C(P_1, P_2)$ 与圆 $C(P_3, P_4)$ 的一个交点, 作另一个交点 P.

3. 结论.

   一个形如 $P' = P''$ 的结论[44], 此处 P' 和 P'' 是由上面两部分定义的两个点.

与前述尺规构图类一样[44], 为了实际构图的方便[44], 可以选取更多的复合作图语句 (比如在 "L 类几何定理证明器"[42] 软件中设计有 27 条作图语句).

§ 43 实例的选取和检验

现在考虑构造性几何命题. 一个几何命题对应于一个代数命题.

已知有三角型方程组 (设系数都是整数)

$$TS: \quad \begin{cases} f_1(u_1, \cdots, u_r; x_1) = 0, \\ f_2(u_1, \cdots, u_r; x_1, x_2) = 0, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ f_s(u_1, \cdots, u_r; x_1, x_2, \cdots, x_s) = 0 \end{cases}$$

(左边的多项式列组成一正常升列), 试问, 在 TS(关于变元 $x_1, \cdots, x_s$) 的所有解中, 有多少个使得多项式方程

$$G: \quad g(u_1, \cdots, u_r; x_1, \cdots, x_s) = 0$$

成立?

依照 §19 所给出的相关性判准, 这等价于问: 在多项式

$$\Phi = \operatorname{res}(g + \lambda, f_s, \cdots, f_1) \in Z[u_1, \cdots, u_r, \lambda]$$

中, λ 的最低次数是多少?

这样, 我们就可以利用 §39 的定理 A 和定理 C, 通过验算一个或多个实例, 来判定 Φ 中所含 λ 的最低次数. 根据定理 A 和定理 C, 现在就只需要分别确定 Φ 关于 $u_1, \cdots, u_r$ 的次数的一个上界了. 由于 Φ 是递归地定义的, 所以这一步很容易做到.

以下总设 Φ 关于 $u_1, \cdots, u_r$ 的次数分别不超过 $n_1, \cdots, n_r$.

几何命题的一个实例, 指的是它的 r 个自由变元 $u_1, \cdots, u_r$ 取一组确定的数值, 然后就这个具体的几何构形来陈述该几何命题. 常常要求这组数值是实数, 以使其确实对应于一个真实的几何构图. 比如, 给定一个三角形 ΔABC, 它的三个顶点的坐标分别为 $A(0,0)$, $B(4,0)$, $C(2,3)$. 就这个三角形而言, 它的三条中线交于一点; 这个事实就是中线定理"三角形三条中线交于一点"的一个实例.

我们说一个实例是一般性的, 如果仅仅通过对它的检验即可判定对应几何命题是否正确.

给定了一个数值实例, 数值检验的办法是: 把这组数值代入 TS, 然后逐个地解出变元 $x_1, \cdots, x_s$, 再把它们代入待检验的方程 G 中, 从而看它是不是成立. 如果是 L 类几何命题, 那么所对应的三角型方程组只有一组解, 这时所要进行的判定和验算就要简单得多.

在确定需要检验的实例之后, 接下来要做的就是对例子的实际检验. 如果这个检验过程只进行整数运算而不涉及浮点计算 (对于

L 类几何命题来说这显然是可以办到的,“L 类几何定理证明器”软件正是据此而编制的), 那么这个过程是毫无疑义的. 可是, 当检验中牵涉有浮点计算时, 就会产生误差, 如何推断最终算出的近似结果是逻辑上绝对的 0 呢? 就是说, 如果最后的等式判断语句中, 等号两边之差非常接近于 0, 那么我们怎么判别这个论断是否正确? 答案还是要从相关性判准得出.

对于一个具体的实例, 如果所有的自由变元都取整数值, 那么多项式

$$\Phi = \operatorname{res}(g+\lambda, f_s, \cdots, f_1) \in Z[u_1, \cdots, u_r, \lambda]$$

就是一个关于的 λ 的整系数多项式, 现在需要判定其中 λ 出现的最低次数. 为简明起见, 让我们考虑 Φ 的常数项, 即当 $u_1, \cdots, u_r$ 取整数值时判定

$$\Phi = \operatorname{res}(g, f_s, \cdots, f_1) \in Z[u_1, \cdots, u_r]$$

是否为 0. 如果 $\Phi \neq 0$, 则 $|\Phi| \geq 1$.

我们可以换个角度来看这个事实, 由 §15 (16.1) 式知

$$\Phi = h_0 g + h_1 f_1 + h_2 f_2 + \cdots + h_s f_s,$$

此时 $f_1 = 0, f_2 = 0, \cdots, f_s = 0$, 所以有 $|h_0 g| = |\Phi| \geq 1$, 即

$$|g| \geq \frac{1}{h_0}.$$

这就是说, 如果结论多项式不等于 0, 则它的值就不会太小; 在 0 与非 0 之间有一个“裂缝”.

如果所取的实例, 其自由变元的取值不是整数而是一些代数数, 比如在单例实验法中自由变元的取值为

$$u_1 = p_1^{1/(n_1+1)},\ u_2 = p_2^{1/(n_2+1)},\ \cdots,\ u_r = p_r^{1/(n_r+1)}.$$

其中 $p_1, p_2, \cdots, p_r$ 是互不相同的素数. 这时不能直接利用上面有关“裂缝”的分析结果. 考虑三角形方程组

$$u_1^{n_1+1} - p_1 = 0,\ u_2^{n_2+1} - p_2 = 0,\ \cdots,\ u_r^{n_r+1} - p_r = 0.$$

与多项式方程 $\Phi=\Phi(u_1,u_2,\cdots,u_r)=0$ 的关系；把这些 p_1,p_2,p_r 看作自由变元，且它们各自的取值就是自身（显然是整数值），就又回到了上面所论情形，有关“裂缝”的分析因而得以保持.

所谓几何定理机器证明的 **数值并行法** 是指，对于给定的几何命题，由计算机自动地确定相应的 $n_1,\cdots,n_r$，并找到满足定理 A 的

$$n=(n_1+1)\times(n_2+1)\times\cdots\times(n_r+1)$$

组数，即 n 个实例，然后对每个实例进行检验，从而判定该几何命题是否成立.

所谓几何定理机器证明的 **单例实验法** 是指，对于给定的几何命题，由计算机自动地确定相应的 $n_1,\cdots,n_r$，然后就自由变元对应取

$$p_1^{1/(n_1+1)},\ p_2^{1/(n_2+1)},\ \cdots,\ p_r^{1/(n_r+1)}$$

的实例（不难证明这个实例是一般的、非退化的）进行检验，从而判定该几何命题是否成立.

§44 例 子

现在我们来看两个例子，看看几何定理机器证明的数值方法的某些关键技术是如何实现的.

[例 1] 给定 ΔABC，作正方形 $ACDF$ 、$BCEG$，求证：直线 BD 垂直于直线 AE.

设 A、B、C、D、E 的坐标分别为

$$A(u_1,u_2),\quad B(1,0),\quad C(0,0),\quad D(x_3,x_4),\quad E(x_1,x_2).$$

则得到三角形方程组：

$$\begin{cases} f_1=x_1=0, \\ f_2=x_2^2-1=0, \\ f_3=x_3^2-u_2^2=0, \\ f_4=u_1x_3-u_4x_4=0. \end{cases}$$

求证：

$$g=(u_1-x_1)(x_3-1)+(u_2-x_2)x_4=0.$$

现在我们来看如何估计消去约束变元 x_1, x_2, x_3, x_4 以后，结论多项式 g 中自由变元 u_1, u_2 的次数界限.

- 首先是 g 与 f_4 关于 x_4 做结式 (消去 x_4)，则 g 中有关变元的次数上界为：

$$(u_1,1),\quad (u_2,1),\quad (x_1,1),\quad (x_2,1),\quad (x_3,1).$$

- 然后与 f_3 关于 x_3 做结式 (消去 x_3)，则各变元的次数上界为：

$$(u_1,2),\quad (u_2,4),\quad (x_1,2),\quad (x_2,2).$$

- 接着再与 f_2 关于 x_2 做结式 (消去 x_2)，则各变元的次数上界为：

$$(u_1,4),\quad (u_2,8),\quad (x_1,4).$$

- 最后与 f_1 关于 x_1 做结式 (消去 x_1)，则各自由变元的次数上界为：

$$(u_1,4),\quad (u_2,8).$$

这样按照单例实验法，对应于如下坐标的三角形构形，即可作为该命题的一个一般实例：

$$A(\sqrt[3]{2},\sqrt[3]{17}),\quad B(1,0),\quad C(0,0).$$

让我们用一个极度简化了的例子来说明，为什么以及怎样从近似计算的结果推断一个不"近似"的结论.

[例 2] 代入 $x=\sqrt[3]{2}$ 作近似计算，从而判定多项式

$$e(x)=(x+1)(x-1)-x^2+1$$

是零多项式.

不妨取一个粗糙的浮点数 $x = 1.26 \pm 0.01$ 代入，计算：

$$\begin{aligned}|e_1| &= |e(\sqrt[3]{2})| \\ &= |(1.26 \pm 0.01 + 1) \times (1.26 \pm 0.01 - 1) - (1.26 \pm 0.01)^2 + 1 \\ &= |0 \pm 0.1| \leq 0.1\end{aligned}$$

其中 $\pm\varepsilon$ 表示 ε 是作浮点计算的累积误差的一个上界.

现在取 $x^3 - 2$ 的另两个根

$$\alpha = \sqrt[3]{2}(\frac{-1}{2} + i\frac{\sqrt{3}}{2}), \quad \beta = \sqrt[3]{2}(\frac{-1}{2} - i\frac{\sqrt{3}}{2}).$$

并作与上面类似的计算，可得：

$$|e_2| = |e(\alpha)| \leq 0.1, \quad |e_3| = |e(\beta)| \leq 0.1.$$

假定 $|e_1| = |e(\sqrt[3]{2})| \neq 0$，则可以断定该多项式不是零多项式. 原因是多项式 $x^3 - 2$ 在有理数域上是不可约的，所以它的根 $\sqrt[3]{2}$ 不可能是一个次数低于 3 的非零整系数多项式的根.

这样，$\mathrm{res}(x^3 - 2, e(x), x)$ 就是一非零整数，即

$$|\mathrm{res}(x^3 - 2, e(x), x)| \geq 1.$$

又

$$\mathrm{res}(x^3 - 2, e(x), x) = |e(\sqrt[3]{2})e(\alpha)e(\beta)|,$$

所以应有

$$|e(\sqrt[3]{2})| \geq \frac{1}{|e(\alpha)e(\beta)|} > \frac{1}{0.01} = 100.$$

这与前面的计算结果 $|e(\sqrt[3]{2})| < 0.1$ 矛盾. 故必定有 $e(\sqrt[3]{2}) = 0$，从而知 $e(x) \equiv 0$.

§ 45 通用程序的运行实例

几何定理机器证明的“数值并行法”和“单例实验法”是两种具有良好实践意义的方法，都已编制成了通用软件．数值并行法在单个处理器上的运行记录参见 [44](其并行处理尚有待实现)，以下是单例实验法的一些运行实例．

［例 1］ (西摩松定理)D, A, B, C 是一圆上的四点，由 D 分别向 BC, CA, AB 作垂线，垂足分别为 E, F, G，证明 E, F, G 共线．

单例实验法运行时间： 1 秒 (486DX/33).

［例 2］ (蝴蝶定理)A, D 是圆 O 上的两点，M 是 AD 的中点，过 M 作圆 O 的两条弦，分别交圆 O 于 B, E 和 C, F, BF 和 CE 分别与 AD 相交于 P 和 Q，证明 M 是线段 PQ 的中点．

单例实验法运行时间： 3 秒 (486DX/33); 1 秒 (HP486/100)

［例 3］ (帕斯卡 (B.Pascal) 定理) A, B, C, D, E, F 是一圆上的六点，P 是 AB 和 DE 的交点，Q 是 BC 和 EF 的交点，R 是 CD 和 FA 的交点，证明 P, Q, R 共线．

单例实验法运行时间： 6 秒 (486DX/33); 1 秒 (HP486/100).

［例 4］ (九点圆) D, E, F 分别是线段 BC, CA, AB 的中点，由 A 向 BC 引垂线，L 为垂足，证明 D, E, F, L 共圆．

单例实验法运行时间： 0 秒 (486DX/33).

［例 5］ (欧拉 (L.Euler) 线) O, G, H 分别是一个三角形的外心、重心、垂心，证明 O, G, H 共线．

单例实验法运行时间： 0 秒 (486DX/33).

［例 6］ (西摩松点定理)A, B, C, D 是一圆上的四点，l_1, l_2, l_3, l_4 分别是 A, B, C, D 关于三角形 BCD, CDA, DAB, ABC 的西摩松线，证明 l_1, l_2, l_3, l_4 共点．

单例实验法运行时间： 5 秒 (486DX/33); 2 秒 (HP486/100).

［例 7］ (费尔巴哈 (K.W.Feuerbach) 定理) 三角形的九点圆与其内切圆及旁切圆相切．

单例实验法运行时间： 27 秒 (486DX/33); 11 秒 (HP486/100).

[例 8] A, B, C, D 是一圆上的四点, H_1, H_2, H_3, H_4 分别是三角形 BCD, CDA, DAB, ABC 的垂心, 证明 AH_1, BH_2, CH_3, DH_4 共点 (我们把这个点称为西摩松点, 其原因请参见例 9).

单例实验法运行时间： 3 秒 (486DX/33).

[例 9] A, B, C, D 是一圆上的四点, H_1, H_2 分别是三角形 BCD, CDA 的垂心, AH_1, BH_2 相交于点 P, E, F 分别是过 C 向 AB, AD 所引垂线之垂足, 证明 E, F, P 共线.

单例实验法运行时间： 3 秒 (486DX/33).

第 6 章

多项式方程的判别系统

本章研究单变元方程解的性态. 这是方程组理论不可缺少的一环. 由于方程的解与多项式的根是一回事, 为了简明, 这里只说多项式而不提方程. 多项式根的理论曾经是代数学的“圣盘”. 虽然现在它不再统治代数学, 但是它的重要性仍是无可置疑的. 许多数学问题都归结为确定某一特定多项式的根, 或者归结为了解根集合的某些信息.

我们假定多项式的系数域是实数域或者实有理函数域 (这其实不是本质性的). 对于 5 次以上的多项式, 我们知道是不存在求根公式的, 所以一般的问题是:

- 确定根的重数.
- 确定根的界限.
- 确定实根的个数.
- 更一般地, 确定根的位置.

对熟知结果的介绍不是本章的重点, 相应的选材是简略而有限的; 本章的重点是给出符号系数多项式实 (虚) 根个数 (重数) 的一个优美的显式判定准则[39],[40]; 其他问题的回答将作为其附带结果而给出.

一个著名的需要用显式判定来解的例题是: 为使

$$x^4 + px^2 + qx + r \geq 0 \qquad (\forall x)$$

成立, p 、q 、r 应满足什么样的充分必要条件？

这个问题曾引起广泛兴趣并被近期的许多作者所研究, 参见文献 [1]、[2]、[3]、[4]、[12]、[25]、[29]、[36]、[53] 等. 读者将会看到, 在熟悉本章提供的基本算法之后, 这问题不过是一个简单的练习.

本书不讨论在指定精确度下计算"近似根"的方法, 现代计算科学提供了大量的技术来处理这一课题, 研究此课题会使我们走得太远.

§ 46 多项式的重根

为了明确, 我们列出一些熟知的概念和结果.

设 F 是基域 K 的任一扩域, $c \in F$, 多项式 $f \in K[x]$. 如果 f 可以被 $(x-c)^k$ 整除, 而不可以被 $(x-c)^{k+1}$ 整除, 则称 c 是 f 的 k 重根 (或 k 重零点). k 叫做这个根的 **重数**. 1 重根往往不叫重根而叫做 **单根**.

设

$$f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$$

是域 K 上的一个次数为 n 的多项式, 它的 **导式** 指的是多项式

$$f'(x) = na_0x^{n-1} + (n-1)a_1x^{n-2} + \cdots + a_{n-1}.$$

定理 1 多项式 $f(x) \in K[x]$ 以 $c \in F$ 为重根, 当且仅当 $f(c) = f'(c) = 0$.

定理 2 设 $a_0 \neq 0$, 则多项式 $f(x) \in K[x]$ 有重根, 当且仅当

$$\operatorname{res}(f(x), f'(x), x) = 0.$$

现在假定 K 是一个特征为 0 的域 (比如有理数域, 实数域, 复数域等). 在因式分解式

$$f(x) = \lambda(p_1(x))^{k_1} \cdots (p_r(x))^{k_r} \qquad \lambda \in K \quad (46.1)$$

中的首一不可约多项式 $p_i(x)$, 称为 $f(x)$ 的 k_i **重因子**(类似于 k 重根). 求给定多项式的因式分解一般是相当困难的. 但是有一种基于导式的方法, 用这种方法可以确定在一给定的域 K 上, $f(x)$ 是否有重根.

定理 3 设 $p(x)$ 是多项式 $f(x)\in K[x]$ 的 k 重因子 ($k\ge 1$, $\deg(p(x),x)\ge 1$), 那么 $p(x)$ 是导式 $f'(x)$ 的 $k-1$ 重因子.

如果一个次数 ≥ 1 的多项式 $f(x)$ 有因式分解 (46.1), 则 $f((x)$ 和它的导式 $f'(x)$ 的最大公因式的因式分解是

$$\gcd(f,f')=(p_1(x))^{k_1-1}\cdots(p_r(x))^{k_r-1}.$$

(在本章里, 我们总假定两个多项式的最大公因式是首一的.)

设多项式

$$\Phi(x)=\frac{f(x)}{\gcd(f,f')}=p_1(x)p_2(x)\cdots p_r(x),$$

它与多项式 $f(x)$ 比较, 次数低、没有重因子, 而且有相同的不可约因式.

进一步, 可以求得 $f(x)$ 的不同重数的因式. 设

$$d_0(x)=f(x),$$

并记

$$\begin{array}{lll}
d_1(x)=\gcd(f(x),f'(x)), & d_2(x)=\gcd(f'(x),(f'(x))'), & \cdots;\\
\Phi_1(x)=d_0(x)/d_1(x), & \Phi_2(x)=d_1(x)/d_2(x), & \cdots;\\
\psi_1(x)=\Phi_1(x)/\Phi_2(x), & \psi_2(x)=\Phi_2(x)/\Phi_3(x), & \cdots.
\end{array}$$

则 $\psi_k(x)$ 是 $f(x)$ 的 k 重因子.

设

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$$

是域 K 上的一个次数为 n 的多项式, 如果 $a_0\ne 0$, 则在复数域中它有 n 个根, 记为 (其中重根出现的次数等于其重数):

$$\alpha_1,\quad \alpha_2,\quad \cdots,\quad \alpha_n.$$

用 s_p 来记这些根的 p 次等幂和：

$$s_p = \sum_{j=1}^{n} \alpha_j^p. \qquad p = 0, 1, 2, \cdots, n$$

特别是 $s_0 = n$.

定理 4 设当 $k < 0$ 或 $k > n$ 时, $a_k = 0$, 则

$$a_{k-1}s_1 + a_{k-2}s_2 + \cdots + a_0 s_k = -k a_k. \qquad k = 1, 2, \cdots$$

这就是所谓的 **牛顿公式**.

记号如上. 考虑式

$$\Delta_n = \prod_{1 \leq j < i \leq n} (\alpha_i - \alpha_j),$$

显然, 可把它写成范德蒙特 (A.T.Vandermonde) 行列式

$$\Delta_n = \begin{vmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{vmatrix}.$$

一般,

$$a_0^{2n-2}\Delta_n^2 = a_0^{2n-2} \prod_{1 \leq j < i \leq n} (\alpha_i - \alpha_j)^2$$

称为多项式 $f(x)$(或 n 元组 $\alpha_1, \alpha_2, \cdots, \alpha_n$) 的 **判别式**,记为 $\mathrm{Dis}(f)$. "判别式"这一名字的来源在于这个函数能够把有两个或多个 α_i 相等的情况与它们全不相同的情况区别开来.

定理 5 多项式 f 有重根当且仅当

$$\mathrm{Dis}(f) = (-1)^{\frac{n(n-1)}{2}} a_0^{-1} \mathrm{res}(f, f') = 0.$$

显然有

$$\Delta_n^2 = \Delta_n \cdot \Delta_n^{Tr} = \begin{pmatrix} s_0 & s_1 & \cdots & s_{n-1} \\ s_1 & s_2 & \cdots & s_n \\ \vdots & \vdots & \ddots & \vdots \\ s_{n-1} & s_n & \cdots & s_{2n-2} \end{pmatrix}.$$

§ 47 实根个数的经典判定法

多项式实根个数的判定, 是一个具有传统兴趣和明显实用价值的重要问题. 关于这个问题, 已有了众多的结果. 其中第一个令人满意的结果, 是在 1829 年由斯图姆 (J.Ch.F.Sturm) 给出的. 本节 (不加证明地) 介绍几个关于多项式实根个数的经典判定法, 这些材料都可以在 [30] 的第 5 章中找到.

首先引进一个下面要用到的定义.

设已给出某一组不为零的, 有限个实数的序列, 例如

$$1, \quad 3, \quad -2, \quad 1, \quad -4, \quad -8, \quad -3, \quad 4, \quad 1. \tag{47.1}$$

依次写出这些数的符号 (用 1 或 -1 来表示):

$$1, \quad 1, \quad -1, \quad 1, \quad -1, \quad -1, \quad -1, \quad 1, \quad 1. \tag{47.2}$$

在这个列 (47.2) 中符号变了四次. 因此我们说序列 (47.1) 的变号数为四次. 对于任何没有零的实数的有限序列当然都可以算出它的变号数.

现在来讨论实系数多项式 $f(x)$ 而且假定多项式 $f(x)$ 没有重根, 否则我们可先除以它和它的导式的最大公因式. 一组不为零的, 有限个实系数多项式的序列

$$f_0(x) = f(x), \ f_1(x), \ f_2(x), \ \cdots, \ f_s(x) \tag{47.3}$$

叫做多项式 $f(x)$ 的**斯图姆组**, 如果它们适合下面四个条件:

- 组 (47.3) 中相邻多项式没有公根.

- 最后一个多项式 $f_s(x)$ 没有实根.
- 如果组 (47.3) 中的多项式 $f_k(x), 1 \leq k \leq s-1$, 有实根 α, 那么 $f_{k-1}(\alpha)$ 和 $f_{k+1}(\alpha)$ 反号.
- 如果 α 是多项式 $f(x)$ 的实根, 那么乘积 $f(x)f_1(x)$ 在 $x=\alpha$ 为增函数. 换句话说, 如果 x 经过 α 上增时, 这一乘积从负值变为正值.

每一个多项式都有斯图姆组, 我们稍后将给出; 现在假定 $f(x)$ 有斯图姆组, 则它可以用来求出实根的个数.

如果实数 r 不是已给出的多项式 $f(x)$ 的根, 而 (47.3) 为这一多项式的斯图姆组, 那么取实数组

$$f_0(r),\ f_1(r),\ f_2(r),\ \cdots,\ f_s(r),$$

删去它里面等于零的数, 且以 $N(r)$ 记余下来这组数的变号数; 把 $N(r)$ 叫做当 $x=r$ 时多项式 $f(x)$ 的斯图姆组 (47.3) 的 **变号数**.

定理 1 (**斯图姆定理**) 如果实数 a 和 b, $a<b$, 不是没有重根的多项式 $f(x)$ 的根, 那么 $N(a) \geq N(b)$, 而且差数 $N(a)-N(b)$ 等于多项式 $f(x)$ 在 a 和 b 间的实根个数.

这样一来, 为了确定 $f(x)$ 在 a 和 b 间的实根个数 (注意, $f(x)$ 应当是没有重根的多项式), 只要求出从 a 变到 b 时, 这一多项式 $f(x)$ 的斯图姆组变号数所减少的数目.

有各种方法可用来构成没有重根的多项式 $f(x)$ 的斯图姆组, 我们来说一个最常见的方法. 取 $f_1(x)=f'(x)$, 然后用 $f_1(x)$ 来除 $f(x)$ 且把余式变号, 取为 $f_2(x)$; 一般地, 如果多项式 $f_{k-1}(x)$ 和 $f_k(x)$ 已经求得, 那么 $f_{k+1}(x)$ 是用 $f_k(x)$ 来除 $f_{k-1}(x)$ 所得出的余

式变号后的多项式. 即

$$
\begin{aligned}
&f_0(x) = f(x),\\
&f_1(x) = f'(x),\\
&f_2(x) = -\mathrm{rem}(f_0(x), f_1(x)),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&f_{k+1}(x) = -\mathrm{rem}(f_{k-1}(x), f_k(x)),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&f_s(x) = -\mathrm{rem}(f_{s-2}(x), f_{s-1}(x)) \neq 0,\\
&f_{s+1}(x) = -\mathrm{rem}(f_{s-1}(x), f_s(x)) = 0.
\end{aligned}
$$

上述的斯图姆判定法对于无重根的多项式是一个完备的算法, 它应用于数值系数多项式特别有效. 但对于文字系数 (或称参系数) 多项式而言, 斯图姆算法不是一个 "显式判定" 方法, 也就是说, 它不是直接使用由其系数组成的若干显式表达式来进行判定. 然而这样的判别式至少对 2 次和 3 次方程是众所熟知的.

如果多项式 $f(x)$ 有重根, 我们必须先除以它和它的导式的最大公因式 $\gcd(f(x), f'(x))$. 比较求两个多项式的最大公因式的辗转相除法与上述求一个多项式的斯图姆组的算法, 我们知道 $f_s(x)$ 除以其首项系数 $\mathrm{lcoeff}(f_s(x))$ 就是 $f(x)$ 和 $f'(x)$ 的 (首一) 最大公因式, 即

$$\gcd(f(x), f'(x)) = \frac{f_s(x)}{\mathrm{lcoeff}(f_s(x))}.$$

沿斯图姆组这条线的某些近期工作, 可参看 [26].

下面两个定理, 虽不能给出实根的确实个数, 只是给出这个数目的上限, 但算法简明, 甚为实用.

设 $f(x)$ 是 n 次实系数多项式, 且允许它有重根存在. 讨论它的逐次导式组

$$f(x), f'(x), f''(x), \cdots, f^{(n-1)}(x), f^{(n)}(x), \tag{47.4}$$

它的最后一项显然为

$$f^{n}(x) = n!\ \mathrm{lcoeff}(f(x)),$$

所以它的符号是确定的. 对任意给定的实数 c, 取正数 ε 适当小, 使得在区间 $(c-\varepsilon, c+\varepsilon)$ 中不含有 (46.4) 中任何一个多项式的除 c 以外的其它根, 在 (46.4) 中分别取 $x=c-\varepsilon$ 和 $x=c+\varepsilon$, 得到两个实数序列

$$f(c-\varepsilon), f'(c-\varepsilon), \cdots, f^n(c-\varepsilon)$$

和

$$f(c+\varepsilon), f'(c+\varepsilon), \cdots, f^n(c+\varepsilon),$$

我们把这两个序列的变号数分别记为 $S(c-)$ 和 $S(c+)$.

定理 2 (**傅里叶** (J.Fourier) **定理**). 如果实数 a 和 $b, a<b$, 不是实系数多项式 $f(x)$ 的根, 那么多项式 $f(x)$ 在 a 和 b 间的实根个数 (l 重根以 l 个计算), 等于差 $N(a+)-N(b-)$ 或比这个差少一个正偶数.

应用傅里叶定理到区间 $(0,\infty)$, 就得到:

定理 3 (**笛卡尔符号法则**) 实系数多项式 $f(x)$ 的正根个数 (l 重根以 l 个计算), 等于这个多项式的系数组的变号数或比这个数少一个正偶数.

假定我们知道 $f(x)$ 的所有根都是实的, 那么就有下面的精确结果:

定理 4 如果实系数多项式 $f(x)$ 的根都是实的, 则它的正根个数 (l 重根以 l 个计算), 等于这个多项式的系数组的变号数.

§ 48 多项式的判别矩阵

设

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_n, \tag{48.1}$$

其导式记为

$$f'(x)=0\cdot x^n+na_0x^{n-1}+(n-1)a_1x^{n-2}+\cdots+a_{n-1}.$$

我们把 $f(x)$ 与 $f'(x)$ 的贝佐矩阵称作多项式 $f(x)$ 的 **判别矩阵**, 称为 $\mathrm{discr}(f)$; 而把 $f(x)$ 与 $f'(x)$ 的西尔维斯特矩阵

$$\begin{pmatrix}
a_0 & a_1 & a_2 & \cdots & a_n & & & \\
0 & na_0 & (n-1)a_1 & \cdots & a_{n-1} & & & \\
 & a_0 & a_1 & \cdots & a_{n-1} & a_n & & \\
 & 0 & na_0 & \cdots & 2a_{n-2} & a_{n-1} & & \\
 & & & \cdots & \cdots & & & \\
 & & & \cdots & \cdots & & & \\
 & & & & a_0 & a_1 & \cdots & a_n \\
 & & & & 0 & na_0 & \cdots & a_{n-1}
\end{pmatrix}$$

(第 2 章的写法与此略有区别) 称作多项式 $f(x)$ 的 **第二判别矩阵**, 记为 $\mathrm{Discr}(f)$.

尽管某些计算机代数软件包 (如 Maple) 能够给出贝佐矩阵, 可是我们这里可以直接地以构造性方式来定义多项式 $f(x)$ 的判别矩阵.

定理 1 多项式 $f(x)$ 如 (48.1). 设当 $k<0$ 或 $k>n$ 时 $a_k=0$. 又设

$$c_{ij}=(n-\max(i,j))a_ia_j-\sum_{p=-1}^{\min(i,j)-1}(i+j-2p)a_pa_{i+j-p}.$$

则 $f(x)$ 的判别矩阵

$$\mathrm{discr}(f)=(c_{ij}). \qquad i,j=0,1,\cdots,n-1$$

这里 i,j 分别为矩阵的行指标和列指标.

根据 §49 中有关矩阵的表达式, 这个定理是易证的.

例如, $g_5=x^5+px^3+qx^2+rx+s$, 则

$$\mathrm{discr}(g_5)=\begin{pmatrix}
5 & 0 & 3p & 2q & r \\
0 & -2p & -3q & -4r & -5s \\
3p & -3q & 3p^2-4r & 2pq-5s & pr \\
2q & -4r & 2pq-5s & 2q^2-2pr & qr-3ps \\
r & -5s & pr & qr-3ps & r^2-2qs
\end{pmatrix}.$$

显然, 判别式矩阵 discr(f) 有较低的阶数, 对于实际的计算甚有益处; 而第二判别式矩阵 Discr(f) 结构简明, 易于记忆. 从后文我们将知道, 就我们的主要目的而言, 这两个判别矩阵起着同样的作用.

本节后面的这部分内容, 给出了判定多项式互异实根个数的一个方法; 尽管不是本章的主要内容, 仍是富有趣味的.

以下总设

$$f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$$

且 $a_0 \neq 0$. 用 M_k 表示判别矩阵 discr(f) 的第 k 个顺序主子式所对应的矩阵. 我们有

定理 2 多项式 $f(x)$ 互异实根的个数等于二次型 $X^{Tr}\cdot\mathrm{discr}(f)\cdot X$ 的符号差; $f(x)$ 恰有 k 个互异根的充要条件是

$$\det(M_k) \neq 0, \quad \det(M_j) = 0. \qquad j = k+1, \cdots, n$$

此时, 其互异实根的个数也等于二次型 $Y^{Tr}\cdot M_k\cdot Y$ 的符号差.

这个定理明显是下面两个引理的推论.

用 $\alpha_1, \alpha_2, \cdots, \alpha_n$ 来记 n 次多项式 $f(x)$ 的 n 个根. 用 s_p 来记这些根的 p 次等幂和:

$$s_p = \sum_{j=1}^{n} \alpha_j^p. \qquad p = 0, 1, 2, \cdots, n$$

特别是 $s_0 = n$. 记 $S_k = (s_{i+j}), i, j = 0, 1, \cdots, k-1$, 即

$$S_k = \begin{pmatrix} s_0 & s_1 & \cdots & s_{n-1} \\ s_1 & s_2 & \cdots & s_n \\ \vdots & \vdots & \ddots & \vdots \\ s_{n-1} & s_n & \cdots & s_{2n-2} \end{pmatrix}.$$

引理 1 记号如上. $f(x)$ 恰有 k 个互异根的充要条件是

$$\det(S_k) \neq 0, \quad \det(S_j) = 0. \qquad j = k+1, \cdots, n$$

此时，其互异实根的个数等于二次型 $Y_j^{Tr}\cdot S_j\cdot Y_j$ 的符号差. 其中,

$$Y_j^{Tr}=(y_0,y_1,\cdots,y_{j-1}). \qquad j=k,k+1,\cdots,n$$

证明 无妨设 $\alpha_1,\alpha_2,\cdots,\alpha_k$ 是 $f(x)$ 的 k 个互异根. $\gamma_{k+1},\cdots,\gamma_j$ 是异于 $\alpha_1,\alpha_2,\cdots,\alpha_k$ 且互异的 $j-k$ 个实数. α_l 的重数记为 $\beta_l(l=1,2,\cdots,k)$. 则

$$\Phi=Y_j^{Tr}\cdot S_j\cdot Y_j=\sum_{l=1}^{n}(y_0+\alpha_l\cdot y_1+\cdots+\alpha_l^{j-1}\cdot y_{j-1})^2.$$

引入如下线性变换：

$$\begin{cases} z_l=\sqrt{\beta_l}\cdot(y_0+\alpha_l\cdot y_1+\cdots+\alpha_l^{j-1}\cdot y_{j-1}), & l=1,2,\cdots,k. \\ z_p=y_0+\gamma_p\cdot y_1+\cdots+\gamma_p^{j-1}\cdot y_{j-1}, & p=k+1,\cdots,j. \end{cases} \tag{48.2}$$

如果多项式 $f(x)$ 的根都是实数，那么 (48.2) 是一个实系数线性变换，而且是满秩的. 这样一来，二次型 Φ 就化为个正平方和：

$$\Phi=\sum_{l=1}^{k}z_l^2.$$

如果多项式 $f(x)$ 有虚根 α_l，那么线性型可以写成下面的形状：

$$z_l=r_l+t_l\sqrt{-1},$$

其中 r_l 和 t_l 都是 $y_0,y_1,\cdots,y_{j-1}$ 的实系数线性型. 但在这一情形，多项式 $f(x)$ 有根 α_p 和 α_l 共轭，故 $z_p=r_l-t_l\sqrt{-1}$. 也就是：

$$z_l^2+z_p^2=2r_l^2-2t_l^2.$$

这样一来，Φ 化为标准型后，由 $f(x)$ 的每一对共轭虚根对应得到一个负平方；同时在标准型中恰好有 k 个平方项. 在这里所用的线性变换是满秩的，因为它们是满秩变换之积. 这就证明了引理的所有结论.

引理 2 设记号 M_k, S_k 如前, 则矩阵 M_k 与矩阵 S_k 合同, 此处 $k=1,2,\cdots,n$.

证明 只要证明存在非奇异的下三角矩阵 A, 使得

$$M_n = \mathrm{discr}(f) = A \cdot S_n \cdot A^{Tr}.$$

其中 A^{Tr} 是 A 的转置. 因为这明显地蕴涵 M_k 合同于 S_k, 对于所有的 $k=1,2,\cdots,n$.

设当 $k<0$ 或 $k>n$ 时, $a_k=0$, 用 A 表示 $n\times n$ 矩阵

$$(a_{i-j}),\quad i,j=0,1,\cdots,n-1.$$

即

$$A=\begin{pmatrix} a_0 & 0 & 0 & \cdots & 0 \\ a_1 & a_0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix}.$$

它是非奇异的, 因为 $a_0\neq 0$.

下证 $M_n = A\cdot S_n\cdot A^{Tr}$.

用 (d_{ij}) 表示矩阵 $A\cdot S_n\cdot A^{Tr}$, 即

$$d_{ij}=\sum_{k=0}^{j}\sum_{l=0}^{i} a_{i-l}s_{k+l}a_{j-k}.\qquad i,j=0,1,\cdots,n-1$$

不失一般性, 我们仅考虑 $j\le i$ 的情形.

在下述推演过程中, 要反复用到牛顿公式:

$$a_{k-1}s_1+a_{k-2}s_2+\cdots+a_0s_k=-ka_k.\qquad k=1,2,\cdots$$

这样就有：

$$
\begin{aligned}
d_{ij} &= \sum_{k=0}^{j} a_{j-k}\left(\sum_{l=0}^{i} a_{i-l}s_{k+l}\right) \\
&= a_j(a_is_0 + a_{i-1}s_1 + \cdots + a_0s_i) + \\
&\quad a_{j-1}(a_is_1 + a_{i-1}s_2 + \cdots + a_0s_{i+1}) + \\
&\quad a_{j-2}(a_is_2 + a_{i-1}s_3 + \cdots + a_0s_{i+2}) + \\
&\quad \cdots + \\
&\quad a_0(a_is_j + a_{i-1}s_{j+1} + \cdots + a_0s_{i+j}) \\
&= (n-i)a_ia_j \\
&\quad -(i+1)a_{i+1}a_{j-1} \\
&\quad -(i+2)a_{i+2}a_{j-2} - a_{i+1}a_{j-2}s_1 \\
&\quad -(i+3)a_{i+3}a_{j-3} - a_{i+1}a_{j-3}s_2 - a_{i+2}a_{j-3}s_1 \\
&\quad - \cdots \\
&\quad -(i+j)a_{i+j}a_0 - a_{i+1}a_0s_{j-1} - \cdots - a_{i+j-1}a_0s_1 \\
&= (n-i)a_ia_j \\
&\quad -(i+1)a_{i+1}a_{j-1} + (j-1)a_{i+1}a_{j-1} \\
&\quad -(i+2)a_{i+2}a_{j-2} + (j-2)a_{i+2}a_{j-2} \\
&\quad - \cdots + \cdots \\
&\quad -(i+j)a_{i+j}a_0 + 0 \cdot a_{i+1}a_{j-1} \\
&= (n-i)a_ia_j - (i+j-2(j-1))a_{i+1}a_{j-1} \\
&\quad -(i+j-2(j-2))a_{i+2}a_{j-2} \\
&\quad - \cdots - (i+j-2\times 0)a_{i+j}a_0 \\
&= (n-i)a_ia_j - \sum_{p=-1}^{j-1}(i+j-2p)a_{i+j-p}a_p \\
&= (n-\max(i,j))a_ia_j - \sum_{p=-1}^{\min(i,j)-1}(i+j-2p)a_p \cdot a_{i+j-p} \\
&= c_{ij}.
\end{aligned}
$$

§ 49 两个判别矩阵的关系

现在我们来建立次数相等的两个多项式的贝佐矩阵与西尔维斯特矩阵之间的关系，作为一种特殊情形，这同时给出了多项式的两个判别矩阵之间的关系.

设

$$
\begin{aligned}
&f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n,\\
&g(x) = b_0x^n + b_1x^{n-1} + \cdots + b_n,\\
&A = \text{Sylvester}(f, g, x) = \begin{pmatrix}
a_0 & a_1 & a_2 & \cdots & a_n & & & \\
b_0 & b_1 & b_2 & \cdots & b_n & & & \\
 & a_0 & a_1 & \cdots & a_{n-1} & a_n & & \\
 & b_0 & b_1 & \cdots & b_{n-1} & b_n & & \\
 & & & \cdots & \cdots & & & \\
 & & & \cdots & \cdots & & & \\
 & & & & a_0 & a_1 & \cdots & a_n \\
 & & & & b_0 & b_1 & \cdots & b_n
\end{pmatrix}\\
&B = \text{Bezout}(f, g, x).
\end{aligned}
$$

即 A, B 分别为多项式 f, g 的西尔维斯特矩阵与贝佐矩阵.

记

$$[p, q] = a_p \cdot b_q - b_p \cdot a_q,$$

$$c_{ij} = \sum_{p=0}^{\min(i,j)} [p, i+j+1-p].$$

以下总设 i, j 分别为矩阵的行指标和列指标. 则

$$B = (c_{ij}). \qquad\qquad i, j = 0, 1, \cdots, n-1$$

对任意矩阵 $D = (d_{ij}), i, j = 1, \cdots, n$, 记

$$D(k, t) = (d_{ij}, d_{i,k+t}), \qquad i = 1, \cdots, k,\ j = 1, \cdots, k-1.$$

$$|D(k, t)| = \det(D(k, t)).$$

其中 t 为整数且 $0 \le t \le n-k$. 显然 $|D(k, 0)|$ 即为矩阵 D 的 k 阶顺序主子式.

本节的主要结论是

定理 1 设矩阵 A, B 如上. 则

$$|A(2k, t)| = |B(k, t)|,$$

其中 $1 \le k \le n$, $0 \le t \le n-k$.

证明 设 $l<0$ 或 $l>n$ 时, $a_l=b_l=0$. 记

$$A_{11}=(a_{j-i}),\qquad i,j=0,1,\cdots,k-1$$

$$=\begin{pmatrix} a_0 & a_1 & \cdots & \cdots & a_{k-1} \\ & a_0 & a_1 & \cdots & a_{k-2} \\ & & \cdots & & \\ & & & & a_0 \end{pmatrix}$$

$$A_{12}=(a_{j+k-i},a_{2k-1-i+t})$$

$$i=0,1,\cdots,k-1;j=0,1,\cdots,k-2$$

$$=\begin{pmatrix} a_k & a_{k+1} & \cdots & a_{2k-2} & a_{2k-1+t} \\ a_{k-1} & a_k & \cdots & a_{2k-3} & a_{2k-2+t} \\ & & \cdots & & \\ a_1 & a_2 & \cdots & a_{k-1} & a_{k+t} \end{pmatrix},$$

$$A_{21}=(b_{j-i}),\qquad i,j=0,1,\cdots,k-1$$

$$A_{22}=(b_{j+k-i},b_{2k-1-i+t}).$$

$$i=0,1,\cdots,k-1;j=0,1,\cdots,k-2$$

易知

$$|A(2k,t)|=(-1)^{\frac{k(k-1)}{2}}\cdot\begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix}.$$

首先来说明 A_{11} 与 A_{21} 是乘法可交换的:

$$A_{11}\cdot A_{21}=\left(\sum_{l=0}^{k-1}a_{l-i}\cdot b_{j-l}\right)=\left(\sum_{l=i}^{j}a_{l-i}\cdot b_{j-l}\right)$$

$$=\left(\sum_{p=i}^{j}a_{j-p}\cdot b_{p-i}\right)=\left(\sum_{p=0}^{k-1}a_{p-i}\cdot b_{j-p}\right)$$

$$=A_{21}\cdot A_{11}$$

因此

$$|A(2k,t)|=(-1)^{\frac{k(k-1)}{2}}|A_{11}\cdot A_{22}-A_{21}\cdot A_{12}|.$$

记

$$\delta_p^q = \begin{cases} 1, & \text{当 } p = q \text{ 时}, \\ 0, & \text{当 } p \neq q \text{ 时}. \end{cases}$$

$$\delta = \left(\delta_i^{k-j-1}\right), \quad i, j = 0, 1, \cdots, k-1.$$

$$D = \delta \cdot (A_{11} \cdot A_{22} - A_{21} \cdot A_{12}).$$

下证 $D = B(k,t)$, 从而原命题成立. 记

$$s(j) = \begin{cases} t, & \text{当 } j = k-1 \text{ 时}, \\ 0, & \text{当 } j < k-1 \text{ 时}. \end{cases}$$

则

$$\begin{aligned}
D &= \delta \cdot \left(\left(\sum_{l=0}^{k-1} a_{l-i} \cdot b_{j+k-l+s(j)} \right) - \left(\sum_{l=0}^{k-1} b_{l-i} \cdot a_{j+k-l+s(j)} \right) \right) \\
&= \delta \cdot \left(\sum_{l=0}^{k-1} [l-i, j-k-l+s(j)] \right) \\
&= \left(\sum_{q=0}^{k-1} (\delta_i^{k-q-1} \cdot \sum_{l=0}^{k-1} [l-q, j+k-l+s(j)]) \right) \\
&= \left(\sum_{l=0}^{k-1} \sum_{q=0}^{k-1} \delta_i^{k-q-1} \cdot [l-q, j+k-l+s(j)] \right) \\
&= \left(\sum_{l=0}^{k-1} [l-(k-i-1), j+k-l+s(j)] \right) \\
&= \left(\sum_{p=-(k-i-1)}^{i} [p, i=j+1-p+s(j)] \right) \\
&= \left(\sum_{p=0}^{\min(i,j)} [p, i=j+1-p+s(j)] \right) = B(k,t).
\end{aligned}$$

多项式序列

$$P_k(f,g)=\sum_{t=0}^{k}|A(2(n-k),t)|x^{k-t}\qquad k=0,1,\cdots,n-1$$

常常称为 f,g 的 **子结式多项式序列**(参见 §10). 根据 §47 的定理, 它也可用贝佐矩阵来求得:

$$P_k(f,g)=\sum_{t=0}^{k}|B(n-k,t)|x^{k-t},\qquad k=0,1,\cdots,n-1$$

如果 $|A(2(n-j),0)|=0, j=0,1,\cdots,k-1$, 而 $|A(2(n-k),0)|\neq 0$, 那么, $P_k(f,g)$ 就是 f,g 的最大公因式.

多项式 $f(x)$ 的重根完全由 $f(x)$ 与 $f'(x)$ 的最大公因式确定. 因此, 我们把 $f(x)$ 与 $f'(x)$ 的子结式多项式序列 $P_k(f,f')$, $k=0, 1,\cdots,n-1$ 称为 $f(x)$ 的 **重因子序列**, 并把 $P_k(f,f')$ 记为 $\Delta_k(f)$.

记 $D_k(f)=|\mathrm{discr}(f)(k,0)|=|\mathrm{Discr}(f)(2k,0)|$, 则序列

$$D_1(f),\ D_2(f),\ \cdots,\ D_n(f)$$

称为 $f(x)$ 的 **判别式序列**. 这个概念在下一节要用到, 在这里先简单介绍一下, 在下文中我们将对之作进一步的讨论.

设 $f(x)$ 的判别式序列中最后一个非零项为 $D_k(f)$, 则 $f(x)$ 与 $f'(x)$ 的最大公因式为 $\Delta_{n-k}(f)$, 即其重因子序列的第 $n-k$ 项 (参见 §10, 定理 1).

§ 50 判别矩阵与斯图姆组的关系

设

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_n,$$
$$g(x)=0\cdot x^n+b_1x^{n-1}+\cdots+b_n.$$

本节将建立 $f(x)$ 、$g(x)$ 的西尔维斯特矩阵与它们的余式序列之间的联系. 从而, 作为一种特例, 我们也得到了多项式的判别矩阵与斯图姆组的关系.

为方便, 这里写出 $f(x)$ 的斯图姆组的构造算法. 记

$$r_0(x) = f(x),$$
$$r_1(x) = b_1 x^{n-1} + \cdots + b_n.$$

用 $r_1(x)$ 来除 $f(x)$ 且把它的余式变号, 记作 $r_2(x)$:

$$r_2(x) = -(r_0(x) - r_1(x) \cdot q_1(x)).$$

一般地, 如果多项式 r_{k-1} 和 $r_k(x)$ 已经求得, 那么 $r_{k+1}(x)$ 是用 $r_k(x)$ 来除 $r_{k-1}(x)$ 所得出的余式变号后的多项式:

$$r_{k+1}(x) = -(r_{k-1}(x) - r_k(x) \cdot q_k(x)).$$

设

$$s_{-1} = -1,$$
$$s_i = \deg(r_i(x)) - \deg(r_{i+1}(x)) - 1, \qquad i = 0, 1, \cdots$$

$$q_j = \sum_{i=-1}^{j-1} (s_i + 1), \qquad j = 0, 1, 2, \cdots$$

$$A_k = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_n & & & & \\ 0 & b_1 & b_2 & \cdots & b_n & & & & \\ & a_0 & a_1 & a_2 & \cdots & a_n & & & \\ & 0 & b_1 & b_2 & \cdots & b_n & & & \\ & & \cdots & \cdots & & & & & \\ & & \cdots & \cdots & & & & & \\ & & & a_0 & a_1 & a_2 & \cdots & a_n \\ & & & 0 & b_1 & b_2 & \cdots & b_n \end{pmatrix}_{2k \times (n+k)}$$

A_n 即为 $f(x), g(x)$ 的西尔维斯特矩阵, 也记之为 A.

记 $\delta_k = \dfrac{1}{2} \sum_{p=-1}^{k-1} s_p \cdot (s_p + 1)$, $\overline{r}_{-1} = 1$, $\overline{r}_i$ 为 $r_i(x)$的 首项系数. 本节的主要结论是

定理 1 记号如前. 我们有

(i) 当 $m \neq \sum_{i=0}^{k-1}(s_i+1)$ 时,

$$|A(2m,0)| = 0;$$

(ii) 当 $m = \sum_{i=0}^{k-1}(s_i+1)$ 时,

$$|A(2m,0)| = (-1)^{\delta_k} \cdot (\overline{r}_0\overline{r}_1)^{s_0+1} \cdot \overline{r}_1\overline{r}_2)^{s_1+1} \cdots \overline{r}_{k-1}\overline{r}_k)^{s_{k-1}+1},$$

$$r_k(x) = \frac{\overline{r}_k}{|A(2m,0)|} \cdot \sum_{t=0}^{n-m} |A(2m,t)| x^{n-m-t}.$$

证明 设 $r_i(x) = r_{i0}x^d + r_{i1}x^{d-1} + \cdots + r_{id}, (r_{i0} \neq 0)$. 为简化起见, 采用下列记号:

$$\begin{pmatrix} r_i & & & \\ & r_i & & \\ & & \ddots & \\ & & & r_i \end{pmatrix} = \begin{pmatrix} r_{i0} & r_{i1} & \cdots & r_{id} & & & \\ & r_{i0} & r_{i1} & \cdots & r_{id} & & \\ & & \cdots & & & & \\ & & & r_{i0} & r_{i1} & \cdots & r_{id} \end{pmatrix},$$

$$R_i = \begin{pmatrix} r_i & & & & 0 & \cdots & 0 \\ & r_i & & & 0 & \cdots & 0 \\ & & \ddots & & 0 & \cdots & 0 \\ & & & r_i & 0 & \cdots & 0 \end{pmatrix}_{(s_{i-1}+s_i+2)\times(2m-q_i)},$$

$$T_i = \left.\begin{pmatrix} r_i & \cdots & \cdots & & & \\ & \ddots & & \ddots & & \\ & & r_i & \cdots & \cdots & \\ r_{i+1} & & & & & \\ & r_{i+1} & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & & r_{i+1} \end{pmatrix}\right. \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} m-q_{i+1} \text{ 行} \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\\a\end{matrix}}\right\} m-q_i \text{ 行} \end{matrix}$$

这里 $i=0,1,\cdots,m-1$.

为了简便，用下列记法表示从矩阵 M 到 M' 的一个变换：

$$M \xrightarrow{v} M', \quad v \equiv v_1+v_2 \pmod 2,$$

它是由若干初等变换组成的一个复合变换．其中包括 (1) 两行互换位置的变换，作 v_1 次；(2) 将某一行反号 (即乘以 -1) 的变换，作 v_2 次；(3) 第 3 类行变换 (即将某行代之以它与另一行之和)，作任意次．这样就有

$$A_m \xrightarrow{\frac{1}{2}m(m-1)} \begin{pmatrix} r_0 & \cdots & \cdots & \cdots & & & & \\ & \ddots & & & \ddots & & & \\ & & \ddots & & & \ddots & & \\ & & & \ddots & & & \ddots & \\ & & & & r_0 & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & r_1 & & & & \\ 0 & \cdots & 0 & & \ddots & & & \\ \vdots & & \vdots & & & \ddots & & \\ 0 & \cdots & 0 & & & & \ddots & \\ 0 & \cdots & 0 & & & & & r_1 \end{pmatrix}$$

$$= \begin{pmatrix} R_0 & \cdots \\ 0 & T_0 \end{pmatrix};$$

$$T_0 \xrightarrow{0} \begin{pmatrix} & & -r_2 & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & & -r_2 \\ r_1 & \cdots & \cdots & & & \\ & \ddots & & \ddots & & \\ & & \ddots & & \ddots & \\ & & & r_1 & \cdots & \cdots \end{pmatrix} \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} m-q_i \text{ 行} \\ \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} m-q_0 \text{ 行} \end{array}$$

$$\xrightarrow{u_0} \begin{pmatrix} R_1 & \cdots \\ & T_1 \end{pmatrix}, \quad 这里\ u_0 = (m-1)(m+1);$$

$$T_1 \xrightarrow{0} \begin{pmatrix} r_1 & \cdots & \cdots & & & \\ & \ddots & & \ddots & & \\ & & r_1 & \cdots & \cdots & \\ r_2 & & & & & \\ & r_2 & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & r_2 & \end{pmatrix}$$

$$\xrightarrow{0} \begin{pmatrix} & & -r_3 & & \\ & & & \ddots & \\ & & & & -r_3 \\ r_2 & & & & \\ & r_2 & & & \\ & & \ddots & & \\ & & & & r_2 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} m-q_2 \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} m-q_1 \end{matrix}$$

$$\xrightarrow{u_1} \begin{pmatrix} R_2 & \cdots \\ & T_2 \end{pmatrix}, \quad 这里\ u_1 = (m-q_1+1)(m-q_2).$$

设 $q_{k-1} < m < q_k$, 则

$$T_{k-2} \xrightarrow{0} \begin{pmatrix} r_{k-2} & \cdots & & & & \\ & \ddots & & \ddots & & \\ & & r_{k-2} & \cdots & \cdots & \\ r_{k-1} & & & & & \\ & \ddots & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & r_{k-1} & \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} m-q_{k-1} \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \\ a \end{matrix}}\right\} m-q_{k-2} \end{matrix}$$

$$\xrightarrow{0}\left(\begin{array}{cccccc} & & -r_k & & \\ & & & \ddots & \\ & & & & -r_k \\ r_{k-1} & \cdots & \cdots & & \\ & \ddots & & \ddots & \\ & & \ddots & & \ddots \\ & & r_{k-1} & \cdots & \cdots \end{array}\right)\begin{array}{l} \left.\right\} m-q_{k-1} \\ \left.\right\} m-q_{k-2} \end{array}$$

$$\xrightarrow{u}$$

$$\left(\begin{array}{cccccccc} r_{k-1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ & \ddots & & & & & \ddots & \\ & & r_{k-1} & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & 0 & \cdots & 0 & r_k & \\ & & & & \ddots & & \ddots & \ddots \\ & & & & & 0 & \cdots & 0 \quad r_k \end{array}\right)\begin{array}{l} \left.\right\} m-q_{k-1} \\ \left.\right\} m-q_{k-2} \end{array}$$

这里 $u=(m-q_{k-2}+1)(m-q_{k-1})$. 此时 $|A_m(2m,0)|=0$, 即

$$|A(2m,0)|=0.$$

设若 $m=q_k=\sum_{i=0}^{k-1}(s_i+1)$, 则

$$|A_m(2m,0)|=(-1)^{\tau_k}(\overline{r}_0\cdot\overline{r}_1)^{s_0+1}\cdot(\overline{r}_1\cdot\overline{r}_2)^{s_1+1}\cdots(\overline{r}_{k-1}\cdot\overline{r}_k)^{s_{k-1}+1}$$

$$r_k(x)=\frac{\overline{r}_k}{|A_m(2m,0)|}\sum_{t=0}^{n-m}|A_m(2m,t)|x^{n-m-t}.$$

其中

$$\tau_k=\frac{1}{2}m(m-1)+\sum_{l=0}^{k-2}\left(\sum_{i=l}^{k-1}(s_i+1)+1\right)\cdot\left(\sum_{i=l+1}^{k-1}(s_i+1)\right).$$

下面往证 $\tau_k\equiv\delta_k(\bmod 2)$:

$$\frac{1}{2}m(m-1)=\frac{1}{2}\left(\sum_{j=0}^{k-1}(s_i+1)\right)\cdot\left(\sum_{i=0}^{k-1}(s_i+1)-1\right)$$

$$= \delta_k + \sum_{0\le i<j\le k-1} s_i\cdot s_j + (k-1)\sum_{j=0}^{k-1} s_j + \frac{1}{2}k(k-1).$$

另一方面,

$$\sum_{l=0}^{k-2}\left(\sum_{i=l}^{k-1}(s_i+1)+1\right)\cdot\left(\sum_{i=l+1}^{k-1}(s_i+1)\right)$$

$$= \sum_{l=0}^{k-2}(s_l+2+\sum_{i=l+1}^{k-1}(s_i+1))\cdot(\sum_{i=l+1}^{k-1}(s_i+1))$$

$$\equiv \sum_{l=0}^{k-2}(s_l\cdot\sum_{i=l+1}^{k-1}(s_i+1)+(\sum_{i=l+1}^{k-1}(s_i+1))^2)(\mathrm{mod} 2)$$

$$\equiv \sum_{l=0}^{k-2}(s_l\cdot\sum_{i=l+1}^{k-1}(s_i+1)+\sum_{i=l+1}^{k-1}(s_i+1))(\mathrm{mod} 2)$$

$$= \sum_{l=0}^{k-2}\sum_{i=l+1}^{k-1} s_l\cdot s_i + \sum_{l=0}^{k-2}(k-l-1)s_l + \sum_{l=0}^{k-2}\sum_{i=l+1}^{k-1} s_i$$

$$+\sum_{l=0}^{k-2}(k-1-l)$$

$$= \sum_{0\le i<j\le k-1} s_i\cdot s_j + \sum_{l=0}^{k-2}(k-l-1)s_l + \sum_{i=0}^{k-1} i\cdot s_i$$

$$+\frac{1}{2}k(k-1)$$

$$= \sum_{0\le i<j\le k-1} s_i\cdot s_j + (k-1)\sum_{i=0}^{k-1} i\cdot s_i + \frac{1}{2}k(k-1),$$

两式相加就有 $r_k \equiv \delta_k (\mathrm{mod}\ 2)$. 注意到 $|\mathbf{A}_m(2m,t)| = |\mathbf{A}(2m,t)|$, $0\le t\le n-m$. 原命题得证.

在上述过程中, 取 $g(x) = f'(x)$, 就得到 $f(x)$ 的斯图姆组 r_0, $r_1, \cdots, r_k$ 与 $f(x)$ 的重因子序列 $\Delta_0(f), \Delta_1(f), \cdots, \Delta_{n-1}(f)$ 以及判别式序列 $D_1(f), D_2(f), \cdots, D_n(f)$ 的如下关系:

(1) 当对于 $j=1,\cdots,k$ 都有 $m\neq q_j=\sum_{i=0}^{j-1}(s_i+1)$ 时, 则有 $D_m(f)=0$. 即, 在判别式序列中, 在 $D_{q_{i-1}}$ 与 D_{q_i} 之间的所有的项都是 0.

(2) 当 $m=q_j=\sum_{i=0}^{j-1}(s_i+1)$ 对某个 $j\,(1\leq j\leq k)$ 成立时,

$$D_m(f)=(-1)^{\delta_j}(\overline{r}_0\cdot\overline{r}_1)^{s_0+1}\cdot(\overline{r}_1\cdot\overline{r}_2)^{s_1+1}\cdots(\overline{r}_{j-1}\cdot\overline{r}_j)^{s_{j-1}+1},$$

$$r_j(x)/\overline{r}_j=\Delta_{n-m}/D_m(f).$$

令 $\sigma_i=D_{q_i}(f)$, 即 σ_i 是 $D_1(f),\cdots,D_n(f)$ 中第 i 个非 0 项, 则

$$\frac{\sigma_{i+1}}{\sigma_i}=(-1)^{s_i(s_i+1)/2}(\overline{r}_i\cdot\overline{r}_{i+1})^{s_i+1}.$$

§ 51 参系数多项式实根个数的显式判定

斯图姆定理完全解决了关于多项式实根个数的问题. 如果所要判定的多项式具有常系数 (有理数、实浮点数等), 斯图姆算法常常是较为有效的. 可是, 当系数中带有参数的时候, 运用斯图姆算法 (或者别的方法) 是琐碎而不切实际的. 比如, 给定一个一般的 n 次多项式

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_n,$$

试给出关于这些文字系数 $a_0,a_1,\cdots,a_n$ 的显式判定条件, 使得对于任给的整数 $k(0\leq k\leq n)$, 多项式 $f(x)$ 恰好有 k 个实根. 这样的条件, 当 $n=4$ 时已经不平凡 (参见 [2]); 对于 $n\geq 5$ 的情形, 则迄今未见有人给出.

另一方面, 当参系数多项式的次数略高时, 在计算机上产生斯图姆序列是很困难的. 当我们试图在一台 586/75(内存 16 兆) 上产生一般 7 次 (参系数) 多项式的斯图姆序列时, 机器运行 1000 多秒之后溢出.

然而, 理论和实践中的大量的判定、设计问题常常牵涉有参数. 例如, 在定理机器证明和发现中, 有关实的代数、几何定理, 不可

避免地要判定带有参数的多项式有无实根；在自动控制中，在机器手工作空间的确定及其设计中，也常常遇到类似的情形．因此，寻求参系数多项式实根个数的简明的显式判准（而不仅仅是"线上"算法）是一件很有意义的工作．

本节给出了一个关于参系数多项式实根个数的简明而完全的显式判定算法．实际的内容更丰富些：我们提出了一个称之为多项式的判别矩阵的概念以及一个由它产生的多项式的判别式序列的概念．在此基础上，给出了多项式判别式序列的所谓符号修订表的变号数与多项式互异实根个数之间的对应关系，给出了实根重数的显式判准以及在给定（有限或无限）区间内多项式实根个数的显式判准．

所有这些表达式及规则，都甚为简明易记，让人过目不忘．

首先重申一下本节要用到的几个定义．

设 $f(x)$ 是一个 n 次多项式． $f(x)$ 的 **判别式序列**

$$D_1(f),\ D_2(f),\ \cdots,\ D_n(f)$$

就是 $f(x)$ 的判别矩阵 $\mathrm{discr}(f)$ 的各阶顺序主子式序列（也就是 $f(x)$ 的第二判别矩阵 $\mathrm{Discr}(f)$ 的偶阶顺序主子式序列）；$f(x)$ 的 **重因子序列**

$$\Delta_0(f),\ \Delta_1(f),\ \cdots,\ \Delta_{n-1}(f)$$

就是 $f(x)$ 与 $f'(x)$ 的子结式多项式序列．

符号修订规则：

接下来我们引进一个新的定义．

对于给定的有限个实数的序列 $l_1,l_2,\cdots,l_n(l_1\neq 0)$．写出相应于这个序列的符号列

$$s_1=\mathrm{sign}(l_1),\ s_2=\mathrm{sign}(l_2),\ \cdots,\ s_n=\mathrm{sign}(l_n).$$

将 $[s_1,\ s_2,\ \cdots,\ s_n]$ 叫做原序列的 **符号表**．

根据这个符号表我们可以定义一个 **符号修订表**

$$[\varepsilon_1,\ \varepsilon_2,\ \cdots,\ \varepsilon_n],$$

其构造规则如下：

- 如果 $[s_i,\ s_{i+1},\ \cdots,\ s_{i+j}]$ 是所给符号表中的一段, 并有 $s_i \neq 0$; $s_{i+1} = s_{i+2} = \cdots = s_{i+j-1} = 0$; $s_{i+j} \neq 0$, 则将此段中由 0 元素构成的序列

$$[s_{i+1},\quad s_{i+2},\quad \cdots,\quad ,s_{i+j-1}]$$

  代之以项数相同的下述序列：

$$[-s_i,\ -s_i,\quad s_i,\quad s_i,\ -s_i,\ -s_i,\quad s_i,\quad s_i,\ -s_i,\ \cdots].$$

  换一种较为形式化却未必直观的说法, 就是令

$$\varepsilon_{i+r} = (-1)^{\left[\frac{r+1}{2}\right]} \cdot s_i. \qquad r = 1, 2, \cdots, j-1$$

- 除此之外, 令 $\varepsilon_k = s_k$, 即, 其余各项保持不变.

例如, 按照上述规则将符号表

$$[1, -1, 0, 0, 0, 0, 0, 1, 0, 0, -1, -1, 1, 0, 0, 0]$$

修订后得到之符号修订表为

$$[1, -1, 1, 1, -1, -1, 1, 1, -1, -1, -1, -1, 1, 0, 0, 0].$$

这个“符号修订规则”其实是很容易记的. 略为细心一点即可发现, 关键在于记住一个数列

$$-1,\quad -1,\quad 1,\quad 1,\quad -1,\quad -1,\quad 1,\quad 1,\quad -1,\quad -1,\quad \cdots.$$

记住这样一个简单的数列, 再记住“第二判别矩阵”的定义 (也很简单, 见 §48), 对整个判准就确能做到“过目不忘”.

在下面将要叙述的本节主要结果中, 我们给出了一个多项式实根个数的计算公式, 它是用该多项式的判别式序列来表达的; 同

时我们也给出了一个更便于记忆和使用的公式，它就是用多项式判别式序列的符号修订表的变号数来表达的.

本节的主要结果是

定理 1 设 $\sigma_0 = 1, \sigma_i = D_{q_i}$ 是 n 次多项式 $f(x)$ 的判别多项式序列 $D_1(f), D_2(f), \cdots, D_n(f)$ 中第 i 个不等于零的项 $(i = 1, \cdots, k)$. 又设 $s_i = q_{i+1} - q_i - 1 (i = 0, 1, \cdots, k-1; q_0 = 0)$, 判别式序列的符号修订表的变号数为 v. 如果 $D_l(f) \neq 0, D_m(f) = 0 (m > l)$, 那么

(i) $f(x)$ 的互异共轭虚根对的数目为 v,

(ii) $f(x)$ 的互异实根的个数为

$$l - 2v = \sum_{i=0,\ s_i \text{偶数}}^{k-1} (-1)^{s_i/2} \cdot \mathrm{sign}(\frac{\sigma_{i+1}}{\sigma_i}).$$

(iii) α 是 $f(x)$ 的 m 重根，当且仅当 α 是 $\Delta_{n-l}(f)$ 的 $m-1$ 重根.

(iv) $D_1(f), \cdots, D_l(f)$ (除去一个正常数因子外) 是无重根多项式 $f/\gcd(f, f')$ 的判别式序列.

证明 对于任意给定的有限个非零实数的序列

$$l_0, l_1, l_2, \cdots, l_n,$$

它的变号数等于序列

$$l_0 l_1, l_1 l_2, \cdots, l_{n-1} l_n$$

中负项的个数, 即

$$\sum_{i=1}^{n} \frac{1}{2} (1 - \mathrm{sign}(l_{i-1} l_i)).$$

现在沿用 §49 的记号, 并引用相应结果, 由于 $D_l(f) \neq 0, D_m(f) = 0 (m > l)$, 所以 $f(x)$ 与 $f'(x)$ 的最大公因式 $\gcd(f, f') = \Delta_{n-l}/D_l(f)$.

这样, 序列

$$R_i(x) = D_l(f) \cdot \frac{r_i(x)}{\Delta_{n-l}(f)}, \qquad (i = 0, 1, \cdots, k)$$

即为 $f(x)/\gcd(f, f')$ 的斯图姆组. 在 $-\infty$ 和 ∞ 处, 其符号如下:

$$\begin{array}{rll} -\infty: & (-1)^{l-q_i} \cdot \mathrm{sign}(D_l(f) \cdot \overline{r}_i, & i = 0, 1, \cdots, k. \\ \infty: & \mathrm{sign}(D_l(f) \cdot \overline{r}_i, & i = 0, 1, \cdots, k. \end{array}$$

根据斯图姆定理, $f(x)$ 的互异实根的个数为

$$\begin{aligned} & \sum_{i=0}^{k-1} \frac{1}{2} \left\{1 - \mathrm{sign}\left((-1)^{2l-q_i-q_{i+1}} \cdot \overline{r}_i \cdot \overline{r}_{i+1} \cdot (D_l(f))^2\right)\right\} \\ & \qquad - \sum_{i=0}^{k-1} \frac{1}{2} \left\{1 - \mathrm{sign}\left(\overline{r}_i \cdot \overline{r}_{i+1} \cdot (D_l(f))^2\right)\right\} \\ = & \sum_{i=0}^{k-1} \frac{1}{2} (1 - (-1)^{s_i+1}) \,\mathrm{sign}\,(\overline{r}_i \cdot \overline{r}_{i+1}) \\ = & \sum_{i=0,\ s_i \text{偶数}}^{k-1} \mathrm{sign}\,(\overline{r}_i \cdot \overline{r}_{i+1}) \\ = & \sum_{i=0,\ s_i \text{偶数}}^{k-1} \frac{1}{2} \cdot \mathrm{sign}\left(\frac{\sigma_{i+1}}{\sigma_i}\right). \end{aligned}$$

设 $f(x)$ 的判别式序列的符号修订表为

$$\varepsilon_1, \ \varepsilon_2, \ \cdots, \ \varepsilon_n$$

其中, 对于所有 $i = 0, 1, \cdots, k-1; p_i = 0, \cdots, s_i; j > q_k$

$$\begin{aligned} & \varepsilon_{q_i+p_i} = (-1)^{p_i(p_i+1)/2} \cdot \mathrm{sign}(\sigma_i), \\ & \varepsilon_{q_k} = \mathrm{sign}(\sigma_k) = \mathrm{sign}(D_l(f)), \\ & \varepsilon_j = 0 \quad (j > q_k). \end{aligned}$$

这个序列的变号数为 v, 故

$$
\begin{aligned}
l-2v &= l-2\cdot\sum_{i=0}^{l-1}\frac{1}{2}\left(1-\operatorname{sign}(\varepsilon_i\cdot\varepsilon_{i+1})\right)\\
&= \sum_{i=0}^{l-1}\operatorname{sign}(\varepsilon_i\cdot\varepsilon_{i+1})\\
&= \sum_{i=0}^{k-1}\left(\sum_{j=0,s_i>0}^{s_i-1}\operatorname{sign}(\varepsilon_{q_i+j}\cdot\varepsilon_{q_i+j+1})+\operatorname{sign}(\varepsilon_{q_i+s_i}\cdot\varepsilon_{q_{i+1}})\right)\\
&= \sum_{i=0}^{k-1}\left(\sum_{j=0,s_i>0}^{s_i-1}(-1)^{\frac{j(j+1)}{2}+\frac{(j+1)(j+2)}{2}}\cdot\operatorname{sign}(\sigma_i^2)\right.\\
&\qquad\left.+(-1)^{s_i(s_i+1)/2}\cdot\operatorname{sign}(\sigma_i\cdot\sigma_{i+1})\right)\\
&= \sum_{i=0}^{k-1}\left(\sum_{j=0,s_i>0}^{s_i-1}(-1)^{j+1}\right.\\
&\qquad\left.+(-1)^{s_i(s_i+1)}\cdot\operatorname{sign}((\overline{r}_i\cdot\overline{r}_{i+1})^{s_i+1}\cdot\sigma_i^2)\right)\\
&= \sum_{i=0}^{k-1}\left(\frac{1}{2}((-1)^{s_i}-1)+\operatorname{sign}((\overline{r}_i\cdot\overline{r}_{i+1})^{s_i+1})\right)\\
&= \sum_{i=0,\ s_i\text{偶数}}^{k-1}\operatorname{sign}(\overline{r}_i\cdot\overline{r}_{i+1})\\
&= \sum_{i=0,\ s_i\text{偶数}}^{k-1}(-1)^{s_i/2}\cdot\operatorname{sign}\left(\frac{\sigma_{i+1}}{\sigma_i}\right).
\end{aligned}
$$

这样, 给定一实参系数多项式, 我们能够很容易地用显式写出其重因子序列, 而重因子序列的每一项又有各自的重因子序列; 所有这些多项式 (原给多项式和包含在不同层次的重因子序列中的多项式) 的判别式序列组成了判定原给多项式实虚根个数和重数的一个完备系统, 我们把这个系统称作为原给 **多项式的判别系统**.

设 $f(a)\neq 0, f(b)\neq 0$. 在有限区间 (a,b) 内, 或在一端有限区间 $(-\infty,a),(b,+\infty)$ 内多项式 $f(x)$ 的实根个数, 显然分别地等于如下多项式

$$(x^2+1)^n\cdot f\left(\frac{ax^2+b}{x^2+1}\right),\quad f(a-x^2),\quad f(x^2+b)$$

的非零实根个数的一半, 这个结果是平凡的. 这样, 我们也就给出了诸情况下实根个数的显式判准. 但这些复合多项式的判别式序列的计算复杂度可能要高好几个数量级. 所以, 当 $f(x)$ 的次数较高时, 这样的判准是不合实用的. 目前我们采取的判别方法是:

1. 用一个补充算法直接产生决定多项式正根的数目的判别式序列. (该补充算法将于另文中介绍).

2. 经过平移可将区间 $(-\infty,a)$ 或 $b,+\infty)$ 中实根数的判定归结为正根数的判定, 即上款情况.

3. 算出 $f(x)$ 的所有实根数, 减去它在 $(-\infty,a)$ 和 $(b,+\infty)$ 中的实根数, 即得其在 (a,b) 内的实根数.

§ 52 例 子

为了计算给定多项式的判别矩阵及其顺序主子式 (即该多项式的判别式序列) 作者用 MAPLE 编了一个很短的通用程序, 不妨叫做 **DISCR**.

```
discr:=proc(poly,var)
   local f,g,tt,d,bz,i,ar,j,mm,dd:
       f:=expand(poly):
       d:=degree(f,var):
       g:=tt*var^d + diff(f,var):
       with(linalg):
       bz:=subs(tt=0,bezout(f,g,var)):
       ar:=[ ]:
       for i to d do
```

```
        ar:=[op(ar),row(bz,d+1-i..d+1-i)] od:
        mm:=matrix(ar):
        dd:=[ ]:
        for j to d do
        dd:=[op(dd),det(submatrix(mm,1..j,1..j))]
        od:     dd:=map(primpart,dd)
    end:
```

当我们键入 "discr(f(x),x)", 该程序输出的 "mm" 就是 $f(x)$ 的判别矩阵 discr(f); 而 "dd" 就是 $f(x)$ 的判别式序列 $D_1,\cdots,D_n$, 某些项可能相差一个正的倍数.

先考虑一个较为容易的例子:

[例 1] 设 $f(x)=x^{20}-x+1$, 试求其实根个数.

程序 DISCR 给出 $f(x)$ 的判别式序列的符号表为:

$$[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,1].$$

符号修订表则为:

$$[1,-1,-1,1,1,-1,-1,1,1,-1,-1,1,1,-1,-1,1,1,-1,-1,1].$$

其变号数是 10, 所以 $f(x)$ 有 10 对共轭虚根, 即没有实根.

[例 2] 求下列多项式的实根和虚根的个数和重数:

$$f(x)=x^{18}-x^{16}+2\,x^{15}-x^{14}-x^5+x^4+x^3-3\,x^2+3\,x-1.$$

由程序 DISCR 给出的 $f(x)$ 的判别式序列的符号表是:

$$[1,\,1,-1,-1,-1,\,0,\,0,\,0,-1,\,1,\,1,-1,-1,\,1,-1,-1,\,0,\,0],$$

故其符号修订表为

$$[1,\,1,-1,-1,-1,\,1,\,1,-1,-1,\,1,\,1,-1,-1,\,1,-1,-1,\,0,\,0],$$

它的变号数为 7 且最后两项为 0. 根据我们的判准, $f(x)$ 应有 7 对相异的虚根和两个相异实根; 仍据判准进一步分析 (即讨论 f 与 f' 的最大公因式 x^2-x+1) 可知其中有一对虚根是二重的.

[例 3] 设二元多项式

$$g(x,y)=x^6-x^4y^2-x^2y^4+y^6-x^4+3x^2y^2-y^4-x^2-y^2+1,$$

求证 $(\forall x\forall y)\quad g(x,y)\geq 0$, 即它是正半定的.

这问题曾被不同作者以不同方法讨论过[56]. 我们的解法是: 先将 $g(x,y)$ 看作是 y 的多项式而 x 作为参数, 这时程序 DISCR 给出的判别式序列 (经因式分解后) 为

$$\begin{aligned}
&1,\quad x^2+1,\quad \left(x^2+1\right)\left(4x^4-7x^2+4\right),\\
&-\left(8x^4-9x^2+8\right)\left(4x^4-7x^2+4\right)(x-1)^2(x+1)^2,\\
&x^2\left(32x^4-61x^2+32\right)\left(8x^4-9x^2+8\right)(x-1)^4(x+1)^4,\\
&-x^4\left(x^2+1\right)\left(32x^4-61x^2+32\right)^2(x-1)^6(x+1)^6.
\end{aligned}$$

易见 $4x^4-7x^2+4$, $8x^4-9x^2+8$, $32x^4-61x^2+32$ 都没有实根从而是严格正定的. 这样, 当 $x\neq 0, x\neq\pm 1$ 时, 该判别式序列的符号表是

$$[1,\ 1,\ 1,\ -1,\ 1,\ -1],$$

其变号数为 3, 这时 $g(x,y)$ 作为 y 的多项式没有实根. 即当 $x\neq 0$, $x\neq\pm 1$ 时, 必有 $g(x,y)>0$, 从而

$$(\forall x\forall y)\quad g(x,y)\geq 0.$$

[例 4]

$$(\forall x)\quad x^6+ax^2+bx+c\geq 0,\tag{52.1}$$

试求参数 a,b,c 所应满足的条件.

设 $f(x)=x^6+ax^2+bx+c$, 则上述问题等价于求 $f(x)$ 没有实根或各实根的重数都是偶数的条件. 此时

$$\operatorname{discr}(f)=\begin{pmatrix}
6 & 0 & 0 & 0 & 2a & b\\
0 & 0 & 0 & -4a & -5b & -6c\\
0 & 0 & -4a & -5b & -6c & 0\\
0 & -4a & -5b & -6c & 0 & 0\\
2a & -5b & -6c & 0 & 2a^2 & ab\\
b & -6c & 0 & 0 & ab & b^2-2ac
\end{pmatrix}$$

程序 DISCR 给出 $f(x)$ 的判别式序列如下 (每项可以各自去掉任意的正因子, 不影响结果):

$$1,\ 0,\ 0,\ a^3,\ D_5',\ D_6',$$

其中

$$\begin{aligned}D_5' &= 256a^5+1728a^2c^2-5400ab^2c+1875b^4,\\ D_6' &= -46656c^5-13824a^3c^3+43200a^2b^2c^2-22500ab^4c\\ &\quad -1024a^6c+256a^5b^2+3125b^6.\end{aligned}$$

这样 (52.1) 要成立当且仅当下述条件之一成立

$$\begin{array}{ll}(1) & D_6'<0\wedge D_5'\geq 0,\\ (2) & D_6'<0\wedge a\geq 0,\\ (3) & D_6'=0\wedge D_5'>0,\\ (4) & D_6'=0\wedge D_5'=0\wedge a>0,\\ (5) & D_6'=0\wedge D_5'=0\wedge a<0\wedge E_2'>0,\\ (6) & D_6'=0\wedge D_5'=0\wedge a=0,\end{array}$$

其中 $E_2'=25b^2-96ac$ 是 $\Delta_2(f)=4ax^2+5bx+6c$ 的判别式.

类似于本例, 可以顺利解决本章引言中提到的关于四次多项式保持正半定的条件问题. 作为一个练习, 请读者完成.

[例 5] 一般五次方程

$$g_5=x^5+px^3+qx^2+rx+s$$

的实根个数 (依其重数) 的完整分类表.

不同于上例, 下表中右边花括号内所列是相应情况下各相异实根的重数. 例如 $\{3,1,1\}$ 表示有一个 3 重实根和两个单实根.

$$\begin{array}{lll}(1) & D_5>0\wedge D_4>0\wedge D_3>0\wedge D_2>0, & \{1,1,1,1,1\}\\ (2) & D_5>0\wedge(D_4\leq 0\vee D_3\leq 0\vee D_2\leq 0), & \{1\}\\ (3) & D_5<0, & \{1,1,1\}\end{array}$$

$$
\begin{array}{lll}
(4) & D_5=0\wedge D_4>0, & \{2,1,1,1\} \\
(5) & D_5=0\wedge D_4<0, & \{2,1\} \\
(6) & D_5=0\wedge D_4=0\wedge D_3>0\wedge E_2\neq 0, & \{2,2,1\} \\
(7) & D_5=0\wedge D_4=0\wedge D_3>0\wedge E_2=0, & \{3,1,1\} \\
(8) & D_5=0\wedge D_4=0\wedge D_3<0\wedge E_2\neq 0, & \{1\} \\
(9) & D_5=0\wedge D_4=0\wedge D_3<0\wedge E_2=0, & \{3\} \\
(10) & D_5=0\wedge D_4=0\wedge D_3=0\wedge D_2\neq 0\wedge F_2\neq 0, & \{3,2\} \\
(11) & D_5=0\wedge D_4=0\wedge D_3=0\wedge D_2\neq 0\wedge F_2=0, & \{4,1\} \\
(12) & D_5=0\wedge D_4=0\wedge D_3=0\wedge D_2=0, & \{5\}
\end{array}
$$

其中

$$
\begin{aligned}
&D_2=-p,\\
&D_3=40rp-12p^3-45q^2,\\
&D_4=12p^4r-4p^3q^2+117prq^2-88r^2p^2-40qp^2s+125ps^2\\
&\quad -27q^4-300qrs+160r^3,\\
&D_5=-1600qsr^3-3750ps^3q+2000ps^2r^2-4p^3q^2r^2+16p^3q^3s\\
&\quad -900rs^2p^3+825q^2p^2s^2+144pq^2r^3+2250q^2rs^2\\
&\quad +16p^4r^3+108p^5s^2-128r^4p^2-27q^4r^2+108q^5s\\
&\quad +256r^5+3125s^4-72p^4rsq+560r^2p^2sq-630prq^3s,\\
&E_2=160r^2p^3+900q^2r^2-48rp^5+60q^2p^2r+1500rpsq\\
&\quad +16q^2p^4-1100qp^3s+625s^2p^2-3375q^3s,\\
&F_2=3q^2-8rp.
\end{aligned}
$$

注: 情况 (2) 对应于有一个单实根和四个互异的虚根；情况 (8) 对应于有一个单实根和一对二重的共轭虚根. 单就实根分类来说, 它们是没有区别的.

§ 53 六次多项式根的分类

前面 §52 中给出的显式判准对于所有高次参系数多项式是完备的, 从理论上讲可以 7 次, 8 次, $\cdots$, 一直作下去. 然而随着次数的增加, 有关的判别式序列的计算复杂度增长极快. 对一般 6 次多项式的根的分类, 已不适合 §52 中对 5 次多项式所采用的那种叙述方式. 对于更高次数的情况, 还可以采取所谓 “lazy 策略”, 即对所得的行列式序列不作展开.

设 $g_6=x^6+px^4+qx^3+rx^2+sx+t$. 令 h_6 表示 g_6 与 $g_6'(x)$ 的最大公因式. 又令 k_6 表示 h_6 与 $h_6'(x)$ 的最大公因式.

下面给出 g_6 的实根 (兼及虚根) 依其重数的详尽的分类表. 共分 23 种情况. 这个分类完全由 g_6, h_6 和 k_6 的判别式序列的符号修订表所确定. 这些判别式序列均属于 g_6 的"完全判别系统", 这一判定实际用到 17 个 (以 g_6 的系数 p,q,r,s,t 为变元的) 多项式. 其中 5 个是平凡的 (它们恒为 1).

1. $\{1,1,1,1,1,1\}$, 即 g_6 有 6 个相异实根, 当且仅当其符号修订表为 $[1,1,1,1,1,1]$.

2. $\{1,1,1,1\}$, 即 g_6 有 4 个单实根, 当且仅当其符号修订表为下列之一者:

$[1,-1,-1,-1,-1,-1]$, $[1,1,-1,-1,-1,-1]$, $[1,1,1,-1,-1,-1]$, $[1,1,1,1,-1,-1]$, $[1,1,1,1,1,-1]$.

3. $\{1,1\}$ (只有 2 个单实根) 而且没有虚的重根, 当且仅当其符号修订表为下列之一者:

$[1,-1,1,1,1,1]$, $[1,-1,-1,1,1,1]$, $[1,-1,-1,-1,1,1]$, $[1,-1,-1,-1,-1,1]$, $[1,1,-1,1,1,1]$, $[1,1,-1,-1,1,1]$, $[1,1,-1,-1,-1,1]$, $[1,1,1,-1,1,1]$, $[1,1,1,-1,-1,1]$, $[1,1,1,1,-1,1]$.

4. $\{\ \}$ (无实根) 而且没有虚的重根, 当且仅当其符号修订表为下列之一者:

$[1,-1,1,-1,-1,-1]$, $[1,-1,1,1,-1,-1]$, $[1,-1,1,1,1,-1]$, $[1,-1,-1,1,-1,-1]$, $[1,-1,-1,1,1,-1]$, $[1,-1,-1,-1,1,-1]$, $[1,1,-1,1,-1,-1]$, $[1,1,-1,1,1,-1]$, $[1,1,-1,-1,1,-1]$, $[1,1,1,-1,1,-1]$.

5. $\{1,1,1,1,2\}$, 即 g_6 有 4 个单实根和 1 个 2 重实根, 当且仅当其符号修订表为 $[1,1,1,1,1,0]$.

6. $\{1,1,2\}$, 即 g_6 有 2 个单实根和 1 个 2 重实根, 当且仅当其符号修订表为下列之一者：
$[1,-1,-1,-1,-1,0],[1,1,-1,-1,-1],[1,1,1,-1,-1],[1,1,1,1,-1]$.

7. $\{2\}$ (只有 1 个 2 重实根) 而且没有虚的重根, 当且仅当其符号修订表为下列之一者：

$[1,-1,1,1,1,0],[1,-1,-1,1,1,0],[1,-1,-1,-1,1,0],$
$[1,1,-1,1,1,0],[1,1,-1,-1,1,0],[1,1,1,-1,1,0].$

8. $\{1,1,2,2\}$, 即 g_6 有 2 个单实根和 2 个 2 重实根, 当且仅当其符号修订表为 $[1,1,1,1,0,0]$ 同时 h_6 的符号修订表为 $[1,1]$.

9. $\{1,1,1,3\}$, 即 g_6 有 3 个单实根和 1 个 3 重实根, 当且仅当其符号修订表为 $[1,1,1,1,0,0]$ 同时 h_6 的符号修订表为 $[1,0]$.

10. $\{2,2\}$, 即 g_6 有 2 个 2 重实根, 当且仅当其符号修订表为下列之一：$[1,-1,-1,-1,0,0],[1,1,-1,-1,0,0],[1,1,1,-1,0,0]$, 同时 h_6 的符号修订表为 $[1,1]$.

11. $\{1,3\}$, 即 g_6 有 1 个单实根和 1 个 3 重实根, 当且仅当其符号修订表为下列之一： $[1,-1,-1,-1,0,0],[1,1,-1,-1,0,0]$, $[1,1,1,-1,0,0]$, 同时 h_6 的符号修订表为 $[1,0]$.

12. $\{1,1\}$ (g_6 只有 2 个单实根) 但有虚的重根, 当且仅当其符号修订表为下列之一： $[1,-1,-1,-1,0,0],[1,1,-1,-1,0,0]$,
$[1,1,1,-1,0,0]$, 同时 h_6 的符号修订表为 $[1,-1]$.

13. $\{\ \}$ (g_6 无实根) 但有 2 个 2 重虚根, 当且仅当其符号修订表为下列之一者：
$[1,-1,1,1,0,0],[1,-1,-1,1,0,0],[1,1,-1,1,0,0].$

14. $\{2,2,2\}$, 即 g_6 有 3 个 2 重实根, 当且仅当其符号修订表为 $[1,1,1,0,0,0]$ 同时 h_6 的符号修订表为 $[1,1,1]$.

15. $\{1,2,3\}$, 即 g_6 有 1 个单实根,1 个 2 重实根和 1 个 3 重实根, 当且仅当其符号修订表为 $[1,1,1,0,0,0]$ 同时 h_6 的符号修订表为 $[1,1,0]$.

16. $\{1,1,4\}$, 即 g_6 有 2 个单实根和 1 个 4 重实根, 当且仅当其符号修订表为 $[1,1,1,0,0,0]$ 同时 h_6 的符号修订表为 $[1,0,0]$.

17. $\{4\}$, 即 g_6 有 1 个 4 重实根, 当且仅当其符号修订表为 $[1,-1,-1,0,0,0]$ 或 $[1,1,-1,0,0,0]$, 同时 h_6 的符号修订表为 $[1,0,0]$.

18. $\{2\}$ (g_6 有 1 个 2 重实根) 并有 2 个 2 重虚根, 当且仅当其符号修订表为 $[1,-1,-1,0,0,0]$ 或 $[1,1,-1,0,0,0]$, 同时 h_6 的符号修订表为 $[1,-1,-1]$ 或 $[1,1,-1]$.

19. $\{3,3\}$, 即 g_6 有 2 个 3 重实根, 当且仅当 g_6 的符号修订表为 $[1,1,0,0,0,0]$, h_6 的符号修订表为 $[1,1,0,0]$, 同时 k_6 的符号修订表为 $[1,1]$.

20. $\{2,4\}$, 即 g_6 有 1 个 2 重实根和 1 个 4 重实根, 当且仅当 g_6 的符号修订表为 $[1,1,0,0,0,0]$, h_6 的符号修订表为 $[1,1,0,0]$, 同时 k_6 的符号修订表为 $[1,0]$.

21. $\{1,5\}$, 即 g_6 有 1 个单实根和 1 个 5 重实根, 当且仅当 g_6 的符号修订表为 $[1,1,0,0,0,0]$ 同时 h_6 的符号修订表为 $[1,0,0,0]$.

22. $\{\ \}$ (g_6 无实根) 但有 2 个 3 重虚根, 当且仅当其符号修订表为 $[1,-1,0,0,0,0]$.

23. $\{6\}$, g_6 有 1 个 6 重实根, 当且仅当其符号修订表为 $[1,0,0,0,0,0]$.

按定义, g_6 的判别式序列 $\{D_1, \cdots, D_6\}$ 由 DISCR 程序给出：

$$D_1 = 1, \qquad D_2 = -p,$$

$$D_3 = 24\,rp - 8\,p^3 - 27\,q^2,$$

$$\begin{aligned}D_4 = {} & 32\,p^4r - 12\,p^3q^2 + 96\,p^3t + 324\,prq^2 - 224\,r^2p^2 - 288\,ptr \\ & -120\,qp^2s + 300\,ps^2 - 81\,q^4 + 324\,tq^2 - 720\,qsr + 384\,r^3,\end{aligned}$$

$$\begin{aligned}D_5 = {} & -4\,p^3q^2r^2 - 1344\,ptr^3 + 24\,p^4q^2t + 144\,pq^2r^3 + 1440\,ps^2r^2 \\ & +162\,q^4tp - 5400\,rts^2 + 1512\,prtsq + 16\,p^4r^3 - 192\,p^4t^2 \\ & +72\,p^5s^2 - 128\,r^4p^2 + 256\,r^5 + 1875\,s^4 - 64\,p^5rt + 592\,p^3tr^2 \\ & +432\,rt^2p^2 - 616\,rs^2p^3 - 558\,q^2p^2s^2 + 1080\,s^2tp^2 \\ & -2400\,ps^3q - 324\,pt^2q^2 - 1134\,tsq^3 + 648\,q^2tr^2 \\ & +1620\,q^2s^2r - 1344\,qsr^3 + 3240\,qst^2 + 12\,p^3q^3s - 1296\,pt^3 \\ & -27\,q^4r^2 + 81\,q^5s + 1728\,t^2r^2 - 56\,p^4rsq \\ & -72\,p^3tsq + 432\,r^2p^2sq - 648\,rq^2tp^2 - 486\,prq^3s,\end{aligned}$$

$$\begin{aligned}D_6 = {} & -32400\,ps^2t^3 - 3750\,pqs^5 + 16\,q^3p^3s^3 - 8640\,q^2p^3t^3 \\ & +825\,q^2p^2s^4 + 108\,q^4p^3t^2 + 16\,r^3p^4s^2 - 64\,r^4p^4t - 4352\,r^3p^3t^2 \\ & +512\,r^2p^5t^2 + 9216\,rp^4t^3 - 900\,rp^3s^4 - 17280\,t^3p^2r^2 \\ & -192\,t^2p^4s^2 + 1500\,tp^2s^4 - 128\,r^4p^2s^2 + 512\,r^5p^2t \\ & +9216\,r^4pt^2 + 2000\,r^2s^4p + 108\,s^4p^5 - 1024\,p^6t^3 - 4\,q^2p^3r^2s^2 \\ & -13824\,t^4p^3 + 16\,q^2p^3r^3t + 8208\,q^2p^2r^2t^2 - 72\,q^3p^3str \\ & +5832\,q^3p^2st^2 + 24\,q^2p^4ts^2 - 576\,q^2p^4t^2r - 4536\,q^2p^2s^2tr \\ & -72\,rp^4qs^3 + 320\,r^2p^4qst - 5760\,rp^3qst^2 - 576\,rp^5ts^2 \\ & +4816\,r^2p^3s^2t - 120\,tp^3qs^3 + 46656\,t^3p^2qs - 6480\,t^2p^2s^2r \\ & +560\,r^2qp^2s^3 - 2496\,r^3qp^2st - 3456\,r^2qpst^2 - 10560\,r^3s^2pt \\ & +768\,sp^5t^2q + 19800\,s^3rqpt + 3125\,s^6 - 46656\,t^5 - 13824\,r^3t^3 \\ & +256\,r^5s^2 - 1024\,r^6t + 62208\,prt^4 + 108\,q^5s^3 - 8748\,q^4t^3 \\ & +729\,q^6t^2 + 34992\,q^2t^4 - 630\,prq^3s^3 + 3888\,prq^2t^3 \\ & +2250\,rq^2s^4 - 4860\,prq^4t^2 - 22500\,rts^4 + 144\,pr^3q^2s^2 \\ & -576\,pr^4q^2t - 8640\,r^3q^2t^2 + 2808\,pr^2q^3st + 21384\,rq^3st^2 \\ & -9720\,r^2q^2s^2t - 77760\,rt^3qs + 43200\,r^2t^2s^2 - 1600\,r^3qs^3 \\ & +6912\,r^4qst - 27540\,pq^2t^2s^2 - 27\,q^4r^2s^2 + 108\,q^4r^3t \\ & -486\,q^5str + 162\,pq^4ts^2 - 1350\,q^3ts^3 + 27000\,s^3qt^2.\end{aligned}$$

在表列情况 8 ~ 13 中, h_6 是一个 2 次多项式, 它的判别式序

列经计算 (可由 DISCR 程序给出) 为：

$$E_1^{(2)} = 1,$$

$$\begin{aligned}E_2^{(2)} &= (-4p^3rq + 24p^4s + 48qpr^2 - 172rp^2s - 36qp^2t + 180stp - 27rq^3 \\ &+126q^2ps - 225s^2q - 216trq + 240sr^2)^2 - (-12p^3q^2 + 32p^4r \\ &+96p^3t + 324prq^2 - 224r^2p^2 - 288ptr - 120qp^2s + 300ps^2 - 81q^4 \\ &+324tq^2 - 720qsr + 384r^3)(-4qp^3s + 64p^4t + 48rpsq - 384rp^2t \\ &-20s^2p^2 - 27q^3s + 324q^2tp - 540tsq + 576tr^2).\end{aligned}$$

在上列情况 14 ~ 18 中, h_6 是一个 3 次多项式, 它的判别式序列可由 DISCR 程序给出：

$$E_1^{(3)} = 1,$$

$$\begin{aligned}E_2^{(3)} &= -32rp^5 + 12q^2p^4 + 288p^4t + 96r^2p^3 - 480qp^3s + 36rp^2q^2 \\ &+300s^2p^2 - 864rp^2t + 972q^2tp + 360rpsq - 432r^2q^2 - 1215q^3s,\end{aligned}$$

$$\begin{aligned}E_3^{(3)} &= 656100r^2s^2q^4 - 512r^3p^9 - 2460375q^5s^3 - 14348907q^6t^2 \\ &-1728s^2p^{10} + 373248p^6t^3 + 54000p^5s^4 + 291600p^4s^2t^2 \\ &-99202.2p^3q^4t^2 - 129600p^6s^3q + 144r^2p^8q^2 - 432p^8sq^3 \\ &+5184r^4p^4q^2 - 2519424r^3q^4t + 13824p^8r^2t + 1728p^8rs^2 \\ &+14256p^7s^2q^2 + 13968p^6r^2s^2 - 124416p^7t^2r + 77760p^7s^2t \\ &-4199040pr^2tq^3s + 1296p^6q^3sr + 44064p^7tq^2r - 13536p^7r^2sq \\ &-108864p^8tsq - 11664p^6q^4t - 41472p^6r^3t - 1667952p^6t^2q^2 \\ &+373248p^5t^2r^2 + 64152p^4q^4s^2 - 1119744p^4t^3r \\ &+1259712p^3q^2t^3 + 455625p^2q^2s^4 + 155520p^5r^2tq^2 + 1536r^4p^7 \\ &+1728p^9rsq + 10628820q^5str + 616896p^6rtsq - 11664p^5q^3ts \\ &-84888p^5rq^2s^2 + 12096p^5r^3sq + 583200p^5st^2q - 298080p^5s^2tr \\ &-31104p^4r^2sq^3 - 52488p^4rq^4t + 6018624p^4q^2t^2r \\ &-637632p^4r^2tsq + 178200p^4rs^3q + 272160p^3r^2q^2s^2 \\ &-933120p^3r^3tq^2 + 5161320p^3q^3trs + 729000p^3tqs^3 \\ &-2799360p^3rst^2q + 708588p^2q^5ts - 43740p^2q^4rs^2 \\ &+23328p^2r^3q^3s + 2361960p^2q^3st^2 - 874800p^2q^2rts^2 \\ &-3359232p^2r^2t^2q^2 - 314928p^2r^2q^4t + 17006112pq^4t^2r \\ &-2952450pq^4ts^2 + 1093500pq^3rs^3 - 1195560p^4q^2ts^2 \\ &-1239300p^3q^3s^3.\end{aligned}$$

在上列情况 19. ~ 22. 中，h_6 是一个 4 次多项式，它的判别式序列可由 DISCR 程序给出：

$$E_1^{(4)} = 1, \quad E_2^{(4)} = 27q^2 - 64rp.$$

$$\begin{aligned} E_3^{(4)} = {} & 144r^2q^2 - 405q^3s - 648q^2tp - 512r^3p \\ & +1680rpsq + 1536rp^2t - 1800s^2p^2, \end{aligned}$$

$$\begin{aligned} E_4^{(4)} = {} & 110592t^3p^3 - 103680sp^2t^2q + 14580q^3str - 4050q^2tps^2 \\ & +93312q^2t^2pr + 13500rps^3q + 86400rp^2ts^2 - 57600r^2psqt \\ & +900q^2r^2s^2 - 3456q^2r^3t - 3375q^3s^3 - 19683q^4t^2 \\ & -73728r^2p^2t^2 - 3200r^3ps^2 + 12288r^4pt - 16875s^4p^2. \end{aligned}$$

在上列情况 19. ~ 20. 中，k_6 是一个 2 次多项式，它的判别式序列经计算为：

$$\begin{aligned} F_1 &= 1, \\ F_2 &= -4096rp^2t + 1200s^2p^2 - 160rpsq + 1728q^2tp + 48r^2q^2 \\ & \quad -135q^3s. \end{aligned}$$

作为本节结果的一个直接应用，我们可以轻易地解决下述的尚无人给出显式判定的一个问题：为了使

$$(\forall x) \quad g_6 = x^6 + px^4 + qx^3 + rx^2 + sx + t \geq 0$$

成立，p, q, r, s, t 应满足什么样的充分必要条件？

根据上面的分类表，g_6 没有实根或只有偶重实根 (即 g_6 是正半定) 的充分必要条件是：其判别系统的符号修订表属于上表中第 4, **7**, **10**,**13**, **14**, **17**, **18**, **20**, **22**, **23** 等 10 种情况之一.

§ 54 稳 定 多 项 式

一个实系数的多项式 $f(x)$ 叫做 **稳定的**，如果它的根都位于 (复平面) 左半平面，即

$$f(\lambda) = 0, \lambda = \alpha + i\beta, \quad \Longrightarrow \quad \alpha < 0.$$

这个术语源自微分方程的理论. 在微分方程中有一个关于动力系统 (可以广泛地理解为力学的、物理的、技术的或经济的体系) 在平衡位置附近是渐近稳定的下述判别法. 若 f 是给定的常系数的 n 阶线性微分方程的特征多项式, 则对任一根 λ, 我们有

$$\lim_{t\to+\infty} e^{\lambda t} = 0.$$

这意味着 λ 的实部为负.

这就引出了一类特殊的局部化问题, 叫做 **路西 (Routh)- 赫尔维茨** (A.Hurwitz) **问题**, 它问如何直接从系数去确定多项式是否稳定. 对任意的 n, 这个代数问题是 1895 年解决的. 经典的结果是：

定理 1 (**路西 - 赫尔维茨定理**) 一个实系数多项式

$$h(z) = c_0 z^n + c_1 z^{n-1} + \cdots + c_n, \qquad c_0 > 0,$$

是稳定的, 当且仅当下面的不等式成立：

$$\Gamma_1 > 0,\ \Gamma_2 > 0,\ \cdots,\ \Gamma_n > 0. \tag{54.1}$$

这里 (设当 $j > n$ 时取 $c_j = 0$)

$$\Gamma_k = \begin{vmatrix} c_1 & c_0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ c_3 & c_2 & c_1 & c_0 & 0 & 0 & \cdots & 0 \\ c_5 & c_4 & c_3 & c_2 & c_1 & c_0 & \cdots & 0 \\ c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{2k-1} & c_{2k-2} & c_{2k-3} & c_{2k-4} & c_{2k-5} & c_{2k-6} & \cdots & c_k \end{vmatrix}$$

容易知道, 如果 $h(z)$ 是稳定的, 那么它必然是一些形如 $z+u$ 和 z^2+vz+w 的因子的乘积, 其中 $u>0, v>0, w>0$, 这就意味着稳定多项式的所有系数都是正的：

$$c_1 > 0,\ c_2 > 0,\ \cdots,\ c_n > 0. \tag{54.2}$$

这样一来，条件 (54.2) 对 $h(z)$ 的稳定性是必要的. 尽管对稳定性而言 (54.2) 不是充分的，但是利用条件 (54.2) 可以把 (54.1) 中的行列式减少一半. 这一点很有实用价值，因为计算行列式是一件比较繁重的事情.

[例] 若 $n=2$，则不等式组 $\Gamma_1>0, \Gamma_2>0$ 等价于最简单的不等式组： $c_1>0$ 和 $c_2>0$. 这个结论也可直接从二次方程的求根公式看出来.

若 $n=3$，则判别法化为不等式 $c_1>0, c_2>0, c_3>0$ 和 $c_1c_2>c_3$，因为我们有 $\Gamma_3=c_3(c_1c_2-c_3)$.

在很多情况下我们要判定复系数多项式的稳定性. 设 $h(z)$ 是一个复系数多项式，考虑多项式 $h(i\cdot z)$，我们把它写为形式：

$$h(i\cdot z)=h_1(z)+i\cdot h_2(z),$$

其中

$$h_1(z)=b_0z^n+b_1z^{n-1}+\cdots+b_n,$$
$$h_2(z)=a_0z^n+a_1z^{n-1}+\cdots+a_n,$$

是两个实系数多项式. 如下判别法成立：

定理 2 设

$$M=\begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_n & & & \\ b_0 & b_1 & b_2 & \cdots & b_n & & & \\ & a_0 & a_1 & \cdots & a_{n-1} & a_n & & \\ & b_0 & b_1 & \cdots & b_{n-1} & b_n & & \\ & & & \cdots & \cdots & & & \\ & & & \cdots & \cdots & & & \\ & & & & a_0 & a_1 & \cdots & a_n \\ & & & & b_0 & b_1 & \cdots & b_n \end{pmatrix},$$

则 $h(z)$ 是稳定的，当且仅当 M 的各偶阶顺序主子式都大于 0.

最后，我们介绍关于多项式稳定性的一个近期的工作.

考虑复系数多项式 $f(z)$，我们把它写为形式：

$$f(x+i\cdot y)=f_1(x,y)+i\cdot f_2(x,y),$$

其中 $f_1(x,y)$ 和 $f_2(x,y)$ 都是关于 x,y 的实系数多项式, 把它们看作为 y 的多项式,表示如下:

$$f_1(x,y)=\alpha_0 y^n+\alpha_1 y^{n-1}+\cdots+\alpha_n,$$

$$f_2(x,y)=\beta_0 y^n+\beta_1 y^{n-1}+\cdots+\beta_n,$$

其中 α_j,β_k 都是关于 x 的实系数多项式.

定理 3 [52] 设

$$R(x)=\text{res}(f_1,f_2,y),$$

则 $f(z)$ 是稳定的, 当且仅当 $R(x)$ 的常数项不等于 0 且 $R(x)$ 的所有系数有相同的符号.

这个结果是笛卡尔符号法则和下述定理的一个推论:

定理 4 [52] $f(z)$ 是稳定的, 当且仅当 $R(x)$ 的所有实根都是负的.

最后这个定理的充分性是显然的; 往证必要性: $f(z)$ 稳定, 设若 $R(x)$ 有一个非负实根 x_0. 此时 $f_1(x_0,y)$ 和 $f_2(x_0,y)$ 必有公共根 $y_0,\overline{y_0}$:

$$y_0=a+i\cdot b,\quad \overline{y_0}=a-i\cdot b,$$

其中 a,b 都是实数. 因此

$$f(x_0-b+i\cdot a)=f(x_0+b+i\cdot a)=0,$$

这就是说, 无论 b 的取值如何, $f(z)$ 必定有一根其实部非负; 这与 $f(z)$ 稳定矛盾.

[1] 本书所载属于作者的工作, 是在国家攀登计划, 八六三高技术项目, 国家自然科学基金, 中国科学院专项经费和国际理论物理中心 (ICTP) 的资助下完成的.

附录 A　用 MAPLE 实现的 WR 程序

```
with(linalg):
wr:=proc(flist,vlist,g)
local ff,d,s,s1,s2,f,x,g1,plist,i,j,k,
      pq,p,q,plist,qlist,v1,v2,h:
ff:=[flist]: d:=[ ]: pq:=[ ]:
s1:=1: s2:=1: s:=0: uset:={}:
for i to nops(flist) do h:=flist[i]:
    uset:=uset union indets(h) od:
uset:=uset minus convert(vlist,set): yb:={}:
for i to nops(flist) do
    if degree(flist[i],vlist[i])>1 then
        yb:=yb union {vlist[i]} fi: od:
g1:=lprem(flist,vlist,g):
while s1<=s2 do plist:=ff[s1]:
    if not(nops(uset)=0) then
        v2:=uts(plist,vlist,uset,g1):
        if v2=0 then if s2>1 then yb:={}:
            for i to nops(plist) do
                x:=op(i,vlist): f:=op(i,plist):
                if degree(f,x)>1 then
                    yb:=yb union {x} fi: od:
            v1:=lprem(plist,vlist,g1):
            v2:=res(plist,vlist,v1):
        else v1:=lprem(plist,vlist,g1):
             v2:=res(plist,vlist,v1):
        fi: fi:
    else if s2>1 then yb:={}:
        for i to nops(plist) do
            x:=op(i,vlist): f:=op(i,plist):
            if degree(f,x)>1 then
                yb:=yb union {x} fi: od:
    v1:=lprem(plist,vlist,g1):
```

```
    v2:=res(plist,vlist,v1):
    else v1:=lprem(plist,vlist,g1):
         v2:=res(plist,vlist,v1): fi: fi:
    if (not(v1=0))and(v2=0) then
        print(`The ascending chain can
               be decomposed into...`):
    h:=wdc(plist,vlist,v1):
    p:=h[1]: k:=h[2]: p:=plist[k]:
    x:=vlist[k]: f:=sprem(f,p,x,'h','q'):
    q:=primpart(q,vlist): s2:=s2+1: s:=s+1:
    plist:=subsop(k=p,plist): qlist:=subsop(k=q,plist):
    ff:=subsop(s1=plist,ff): ff:=[op(ff),qlist]:
    pq:=[op(pq),[k,p,q]]; `f`.k.`1`:=p;`f`.k.`2`:=q;
    print(`f`.k.`1:=`(p));print(`f`.k.`2:=`(q));
    if s2>s1+1 then k1:=s1: k2:=s2: fi:
    else print(`****** (`.s1.`) ******`):
        d:=[op(d),[v1,v2]]: h:=[ ]:
        for i to nops(vlist) do k:=0:
            for j to s do if i=op(1,pq[j]) then
                if op(i,plist)=op(2,pq[j]) then
                    h:=[op(h),`f[`.i.`1]`]: k:=i:
                elif op(i,plist)=op(3,pq[j]) then
                    h:=[op(h),`f[`.i.`2]`]: k:=i:
                fi: fi: od:
            if not(k=i) then h:=[op(h),`f[`.i.`]`]
        fi: od:
        print(h):
        if not(op(2,d[s1])=0) then
            print(`The final resultant is not zero.`):
        elif op(1,d[s1])=0 then
            print(`The final pseudo remainder is zero.`)
        fi: s1:=s1+1: fi: od:
[ff]:
end:

lprem:=proc(flist,vlist,g)
local f,h,i,m,n,x,y:
h:=primpart(g,vlist): y:=yb:
```

```
for i from nops(flist) by-1 to 1 do
     f:=flist[i]: x:=vlist[i]:
     n:=degree(f,x): m:=degree(h,x):
     if not(n=0) and not(n>m) then
          if i>1 then h:=prem(h,f,x): x:=vlist[i-1]:
               if degree(h,x)>1 then y:=y union {x}: fi:
               h:=primpart(h,y):
          else h:=prem(h,f,x): h:=primpart(h,vlist):
          fi:fi:od:
h:
end:

res:=proc(flist,vlist,g)
local i,f,h,x:
h:=g:
for i from nops(flist) by-1 to 1 do
     f:=flist[i]: x:=vlist[i]:
     if not(f=0)and(degree(h,x)>0) then
          h:=resultant(h,f,x): h:=lprem(flist,vlist,h):
     fi:od:
h:
end:

subres:=proc(f,g,x,k)
local i,j,t,r,m,n,a,b,s:
if k=0 then s:=resultant(f,g,x):
else n:=degree(f,x): m:=degree(g,x): t:=m+n:
     a:=array(1..t,1..t): a:=sylvester(f,g,x):
     r:=m+n-2*k: b:=array(1..r,1..r):
     for j to t do for i to t do
          if (j<=r)and(i<=m-k) then b[i,j]:=a[i,j]:
          elif (j<=r)and(i<=t-k)and(i>m) then
               b[i-k,j]:=a[i,j]:
          fi: od: od: s:=det(b): fi:
s:
end:

submix:=proc(f,g,x,k,d)
```

```
local i,j,t,r,m,n,a,b,s:
if k=0 then s:=resultant(f,g,x):
else n:=degree(f,x): m:=degree(g,x): t:=m+n:
    a:=array(1..t,1..t): a:=sylvester(f,g,x):
    r:=m+n-2*k: b:=array(1..r,1..r):
    for j to t do if not(j=t) then for i to t do
    if (j<=r-1)and(i<=m-k) then b[i,j]:=a[i,j]:
    elif (j<=r-1)and(i<=t-k)and(i>m) then
        b[i-k,j]:=a[i,j]: fi: od:
    else for i to t do if i<=m-k then
        b[i,r]:=a[i,t-k-d]: elif (i<=t-k)and(i>m) then
        b[i-k,r]:=a[i,t-k-d] fi: od:
    fi: od: s:=det(b): fi:
s:
end:

uts:=proc(plist,vlist,ulist,g)
local m,n,f,flist,h,u,u1,p,i,j,v,r,x:
m:=nops(ulist): n:=nops(plist): f:=array(1..n):
ulist1:=[-1,-2,1,2,-1,0,1,-1,0,1,-1,0,1,
         -1,0,1,-1,0,1,-1,0,1]: h:=g:
for j to m do u:=op(j,ulist):
    u1:=op(j,ulist1): h:=subs(u=u1,h): od:
h:=primpart(h,vlist): r:=1:
for i from n by-1 to 1 do if not(r=0) then
    p:=op(i,plist): if not(p=0) then
        x:=op(i,vlist): for j to m do
        u:=op(j,ulist): u1:=op(j,ulist1):
        p:=subs(u=u1,p): od:
        if not(p=0) then
            r:=prem(1,p,x): f[i]:=primpart(p,vlist):
        else r:=0: fi: else f[i]:=0: fi: fi: od:
if r=0 then h:=0: else flist:=convert(f,list):
    h:=res(flist,vlist,h): fi:
h:
end:

wdc:=proc(flist,vlist,g)
```

```
local f,h,x,hlist,r,s,u,v,t,d,p,q,i,k:
hlist:=flist: h:=g: v:=0:
for i from nops(flist) by-1 to 1 do if v=0 then
    f:=hlist[i]: x:=vlist[i]: n:=degree(f,x):
    m:=degree(h,x): t:=min(m,n):
    hlist:=subsop(i=0,hlist): r:=0:
    for k from 0 to t do if r=0 then s:=subres(h,f,x,k):
        if not(s=0) then p:=lprem(hlist,vlist,s):
            if not(p=0) then d:=k: r:=1: fi: fi: fi: od:
    if not(r=0) then q:=uts(hlist,vlist,uset,p):
        if q=0 then q:=res(hlist,vlist,p): fi:
        if not(q=0) then t:=s*x^d:
            for k from 0 to d-1 do
            t:=t+submix(h,f,x,d,k)*x^k: od:
            t:=[primpart(t,vlist),i]: v:=1:
        else h:=p: fi: fi: fi: od:
t:
end:
```

附录 B　用 MAPLE 实现的 GPS 程序

```
with(linalg):
tsF:=proc(a,b)
local ret, vr, i, loclst;
loclst:=tsvar; ret:=1;
if degree(a)<degree(b) then ret:=true; else i:=1;
    while ((ret = 1) and not(i = 1+nvars)) do
        vr:=loclst[i]; dga:=degree(a,vr);
        dgb:=degree(b,vr); if dga<dgb then ret:=true;
        elif dga > dgb then ret:=false;
        else i:=i+1; fi; od; fi;
if ret=1 then ret := false; fi;
ret;
end:
```

```
delz:=proc(v)
local vv,vvv,i,n;
vv:=convert(v,'list'): n:=nops(vv): vvv:=[ ];
for i to n do if not(vv[i]=0) then
    vvv:=[op(vvv),vv[i]]: fi: od:
end:

readlib(factors):
gps:=proc(tsplst,tsvlst)
st:=time(); npols:=nops(tsplst); nvars:=nops(tsvlst);
tspowers:=array(1..nvars);
if not(npols=nvars+1) then
    print('Number of equations should be number of
           variables plus one'); else
tsA:=array(1..npols,1..npols); tsVr:=array(1..nvars);
for i from 1 to nvars do lcvr:=op(i,tsvlst);
    tsVr[i]:=ts.lcvr.r od; tsVrlst:=convert(tsVr,list);
# Setting up the first row of the matrix.
for i from 1 to npols do tsA[1,i]:=op(i,tsplst); od;
# Setting up the remaining rows of the matrix.
divby:=1; for i from 2 to npols do
    subvar:=op(i-1,tsvlst); subby:=tsVr[i-1];
    for j from 1 to npols do
        tsA[i,j]:=subs(subvar=subby,tsA[i-1,j]);od;od;
for i from npols by -1 to 2 do
    subvar:=op(i-1,tsvlst); subby:=tsVr[i-1];
    for j from 1 to npols do tsA[i,j]:=
        normal((tsA[i,j]-tsA[i-1,j])/(subvar-subby));
od; od;
print('Set up the First Matrix');
#Compute the determinant of the matrix ... "dp".
tsa:=normal(det(tsA));
tsdp:=collect(tsa,tsVrlst,distributed);
print('Evaluated the First Determinant');
# Setting up the actual Dixon Matrix.
tscl:=[coeffs(tsdp,tsVrlst,'trs')];
coeffs(collect(tsdp,tsvlst,distributed),tsvlst,'txy');
```

```
tsvar:=tsvlst; nxy:=nops([txy]);
sxy:=sum(txy[q],q=1..nxy);
txy:=[op(sort(sxy,tsvlst,tdeg))];
nrs:=nops(tscl); tsB:=array(1..nrs,1..nxy);
for i from 1 to nxy do trm:=op(i,txy);
    for j from 1 to nvars do tspowers[j]:=0;
        while divide(trm,tsvlst[j]^(1+tspowers[j])) do
            tspowers[j]:=tspowers[j]+1; od; od;
    for j from 1 to nrs do
        tsB[j,i]:=collect(tscl[j],tsvlst[1]);
        for k from 1 to nvars do
            tsB[j,i]:=collect(tsB[j,i],tsvlst[k]);
            tsB[j,i]:=coeff(tsB[j,i],tsvlst[k],
                                    tspowers[k]);
        od; od; od;
print(`Set up the Dixon Matrix`);
print(`Doing Gaussian Elimination`);
tsC:=ffgausselim(tsB); gps:=multiply(tsC,txy); fi;
gps;
end:
```

注：上述是使用全幂序的 GPS 程序. 若将其第 56 行中的 “tdeg” 更换为 “plex”, 其余不必改动, 就得到使用纯字典序的 GPS 程序.

附录 C 用 MAPLE 实现的 WRSOLVE 程序

```
readlib(factors):
with(linalg):
wrsolve:=proc(fset,var)
local ff,f,x,c,flist,vlist,gset,hset,hlist,
      h,h1,s1,s2,i,j,k,t,b,d,n:
tim0:=time(): results:=[ ]:
ulist0:=[-1,-2,1,2,-1,-2,1,2,-1,-2,1,2,-1,
        -2,1,2,-1,-2,1,2,-1,-2,1,2]:
uset:={}: wset:={}: ff:=[fset]: vset:=var:
```

```
hset:=map(expand,fset):
for i to nops(vset) do x:=op(i,vset):
   for j to nops(hset) do h:=op(j,hset):
      uset:=uset union indets(h): h:=subs(x=0,h):
      if not(h=0) and (indets(h) intersect vset={}) then
         wset:=wset union {x}: fi: od: od:
s1:=1: s2:=1: ff:=[hset]:
while s1<=s2 do hset:=ff[s1]:hlist:=convert(hset,list):
   k:=1: for i to nops(hlist) do h:=hlist[i]:
      if indets(h) intersect vset={} then k:=0: fi: od:
   if k=1 then for i to nops(hset) do h:=hlist[i]:
      if (hastype(h,'+')=false) and (wset intersect
         indets(h)={}) then h:=primpart(h,vset):
         for j to nops(hlist) do if not(i=j) then
            h1:=hlist[j]: h1:=subs(h=0,h1):
            hlist:=subsop(j=h1,hlist): fi: od:
      elif (hastype(h,'+')=false) and
         not(wset intersect indets(h)={}) then
         k:=0: fi: od: fi:
   if k=1 then h:=as(hset,vset): flist:=h[1]:
      vlist:=h[2]: gset:=h[3]:
      if not(gset={}) then t:=tsas(flist,vlist):
         flist:=t[1]:
         if not(t[2]={}) then
            for i to nops(flist) do
               f:=flist[i]:
               h:=indets(f) intersect vset:
               if h={} then k:=0: fi: od:
            if k=1 then h1:=dc(flist,vlist,op(1,t[2])):
               for i to nops(h1) do
                  h:=op(i,h1): s2:=s2+1: ff:=
                  [op(ff),convert(h,set) union gset]:
               od: k:=0: fi: fi: fi: fi:
   if k=1 then if not(gset={}) then g:=op(1,gset):
      m:=nops(flist):
      for j from 2 to nops(gset) do
         if degree(op(j,gset),vset)<degree(g,vset) then
            g:=op(j,gset): fi: od:
```

```
      hset:=uset minus convert(vlist,set):
      hlist:=flist: h:=fw(flist,vlist,g): flist:=h[1]:
      vlist:=h[2]: hset:=uset minus convert(vlist,set):
      h:=wr(flist,vlist,hset,g):
      b:=h[1]: d:=h[2]: n:=h[3]:
      for i to n do
         if not(op(2,d[i])=0) then
           if not(indets(op(2,d[i])) intersect vset={})
              then hset:=gset minus {g}:
              hset:=hset union {op(hlist),op(2,d[i])}:
              s2:=s2+1: ff:=[op(ff),hset]: fi:
         elif op(1,d[i])=0 then
            t:=tsas(b[i],vlist):h:=t[1]:
            if not(t[2]={}) then s2:=s2+1:
              ff:=[op(ff),convert(h,set) union gset]:
            else hset:=gset minus {g}:
              hset:=hset union convert(b[i],set):
              s2:=s2+1: ff:=[op(ff),hset]:
              for j to nops(vlist) do f:=op(j,b[i]):
                x:=op(j,vlist): c:={coeffs(f,x)}: k:=0:
                for k to nops(c) do
                  if k=0 then h:=op(k,c):
                    if indets(h) intersect vset={} then
                      k:=1: fi: fi: od:
                if k=0 then
                  fset:=convert(b[i],set) minus {f}:
                  fset:=fset union c union gset:
                  s2:=s2+1: ff:=[op(ff),fset]:
                fi: od: fi: fi: od:
      ff:=subsop(s1=ff[s2],ff): ff:=subsop(s2=NULL,ff):
      s2:=s2-1:
      else ff:=subsop(s1=flist,ff): s1:=s1+1: fi:
   else ff:=subsop(s1=ff[s2],ff):
      ff:=subsop(s2=NULL,ff): s2:=s2-1: fi: od:
for i to s2 do results:=[op(results),ff[i]]: od:
results:
end:
```

```
as:=proc(fset,vset)
local flist,vlist,m,n,glist,xlist,gset,i,j,f,g,x,d,d1,
      k1,k2,k3,h,hlist:
flist:=convert(fset,list): vlist:=convert(vset,list):
n:=nops(flist): m:=nops(vlist): glist:=[ ]: xlist:=[ ]:
while not(n=0) do if not(n=1) then
   m1:=array(1..n,1..m): m2:=array(1..m):
   m3:=array(1..n): m4:=array(1..n):
   for i to n do m3[i]:=0: for j to m do
      m1[i,j]:=0: m2[j]:=0: od: od:
   for i to n do f:=op(i,flist): m4[i]:=degree(f,vset):
      for j to m do x:=vlist[j]: d:=degree(f,x):
         if not(d=0) then m1[i,j]:=d: m2[j]:=m2[j]+1:
            m3[i]:=m3[i]+1: fi: od: od:
   k1:=n: for j to m do if not(m2[j]=0) then
     if m2[j]<k1 then k1:=m2[j]: fi: fi: od:
   k2:=0: for j to m do if m2[j]=k1 then for i to n do
     if not(m1[i,j]=0) then if m3[i]>k2 then k2:=m3[i]:
     g:=flist[i]: x:=vlist[j]: d:=m1[i,j]: k3:=j:
     h:=m4[i]: fi: fi: od: fi: od:
   for j to m do if m2[j]=k1 then for i to n do
     if (m3[i]=k2)and(not(m1[i,j]=0)) then
       if m1[i,j]<d then g:=flist[i]: x:=vlist[j]:
         k3:=j: d:=m1[i,j]: h:=m4[i]:
       fi: fi: od: fi: od:
   for j to m do if m2[j]=k1 then for i to n do
     if (m3[i]=k2)and(m1[i,j]=d) then if m4[i]<h then
       h:=m4[i]: g:=flist[i]: x:=vlist[j]: k3:=j:
       fi: fi: od: fi: od:
   hlist:=[ ]: for i to n do if m1[i,k3]=0 then
     hlist:=[flist[i],op(hlist)]: fi: od:
   flist:=hlist: vlist:=subsop(k3=NULL,vlist):
   glist:=[g,op(glist)]: xlist:=[x,op(xlist)]:
   else g:=flist[1]: glist:=[g,op(glist)]: k1:=0:
     for j to m do x:=op(j,vlist): d:=degree(g,x):
       if k1=0 then if not(d=0) then
         d1:=d: k1:=1: k2:=j: fi:
       elif (d<d1)and(not(d=0)) then d1:=d: k2:=j:
```

```
        fi: od:
      xlist:=[op(k2,vlist),op(xlist)]: flist:=[ ]:
    fi: n:=nops(flist): m:=nops(vlist): od:
gset:=fset minus convert(glist,set):
[glist,xlist,gset]:
end:

tsas:=proc(flist,vlist)
local f,h,t,n,u,i,j,k1,k2,c,cj,x,wlist,hlist:
t:={}: hlist:=[flist[1]]:
wlist:=[vlist[1]]: n:=nops(flist):
for i from 2 to n do if t={} then
  f:=flist[i]: x:=vlist[i]: c:=lcoeff(f,x): k1:=0:
  u:=uset minus convert(wlist,set):
  k2:=uts(hlist,wlist,u,c):
  if not(k2=0) then k1:=1: fi:
  if k1=0 then k2:=res(hlist,wlist,c):
    if not(k2=0) then k1:=1: fi: fi:
  if k1=0 then h:=lprem(hlist,wlist,f):
    hlist:=subsop(i=h,flist): t:={i}:
  else hlist:=[op(hlist),f]: wlist:=[op(wlist),x]:
  fi: fi: od:
[hlist,t]:
end:

dc:=proc(flist,vlist,n)
local g,x,c,hset,hlist,wlist,i,k,h,h1,b,d,m:
g:=flist[n]: x:=vlist[n]: hset:={}:
hlist:=flist: wlist:=vlist: h1:=hlist:
for i to n do h1:=subsop(1=NULL,h1): od:
  for i to nops(h1)+1 do
    hlist:=subsop(n=NULL,hlist):
    wlist:=subsop(n=NULL,wlist): od:
  if not(degree(g,x)=0) then c:=lcoeff(g,x):
    h:=uset minus convert(vlist,set):
    h:=wr(hlist,wlist,h,c):
    b:=h[1]: d:=h[2]: m:=h[3]:
    for k to m do if op(2,d[k])=0 then
```

```
      h:=lprem(b[k],wlist,g):
      if not(indets(h) intersect vset={}) then
        if h=0 then
          hset:=hset union {[op(b[k]),op(h1)]}:
        else hset:=hset union {[op(b[k]),h,op(h1)]}:
        fi: fi:
  else hset:=hset union {[op(b[k]),g,op(h1)]}: fi: od:
  else hset:=hset union {flist}: fi:
hset:
end:

fw:=proc(flist,vlist,g)
local i,j,k,s,r,x,h,hset,hlist,wlist:
hset:=vset minus convert(vlist,set):
hset:=indets(g) intersect hset: s:=0:
for i to nops(hset) do x:=op(i,hset): r:=0:
for j to nops(flist) do h:=flist[j]:
  if member(x,indets(h))and(r=0) then k:=j-1: r:=1:
  fi: od:
  if not(k<s) then s:=k: fi: od:
hlist:=flist: wlist:=vlist:
if not(s=0) then for i to s do
  hlist:=subsop(1=NULL,hlist):
  wlist:=subsop(1=NULL,wlist): od: fi:
[hlist,wlist]:
end:

wr:=proc(flist,vlist,par,g)
local ff,d,s,s1,s2,f,x,g1,plist,i,j,k,
      p,q,plist,qlist,v1,v2,h:
ff:=[flist]: d:=[ ]: s1:=1: s2:=1:
g1:=lprem(flist,vlist,g):
while s1<=s2 do plist:=ff[s1]:
  v1:=lprem(plist,vlist,g1): v2:=res(plist,vlist,v1):
  if (not(v1=0))and(v2=0) then
    print('The ascending chain can be
           decomposed into...'):
    h:=wdc(plist,vlist,v1): p:=h[1]: k:=h[2]:
```

```
      f:=plist[k]: x:=vlist[k]: f:=sprem(f,p,x,'h','q'):
      q:=primpart(q,vlist): s2:=s2+1:
      plist:=subsop(k=p,plist): qlist:=subsop(k=q,plist):
      ff:=subsop(s1=plist,ff): ff:=[op(ff),qlist]:
    else d:=[op(d),[v1,v2]]: s1:=s1+1: fi: od:
  s1:=s2: for i to s2 do
    if not(indets(op(2,d[i])) intersect vset={}) then
      h:=op(2,factors(op(2,d[i]))):
      for j to nops(h) do f:=op(1,op(j,h)):
        if hastype(f,`+`)=true then s2:=s2+1:
        ff:=[op(ff),ff[i]]: d:=[op(d),[1,f]]:
        elif indets(f) intersect wset={} then s2:=s2+1:
          ff:=[op(ff),ff[i]]: d:=[op(d),[1,f]]: fi: od:
        else s2:=s2+1: ff:=[op(ff),ff[i]]:
        d:=[op(d),d[i]]: fi: od:
     for i to s1 do ff:=subsop(1=NULL,ff):
       d:=subsop(1=NULL,d): od:
     s1:=s2-s1:
  [ff,d,s1]:
  end:

lprem:=proc(flist,vlist,g)
local f,h,i,m,n,x:
h:=primpart(g,vset):
for i from nops(flist) by-1 to 1 do
  f:=flist[i]: x:=vlist[i]:
  n:=degree(f,x): m:=degree(h,x):
  if not(n=0) and not(n>m) then
    h:=prem(h,f,x): h:=primpart(h,vset): fi: od:
h:
end:

res:=proc(flist,vlist,g)
local i,f,h,x:
h:=g:
for i from nops(flist) by-1 to 1 do
  f:=flist[i]: x:=vlist[i]:
  if not(f=0)and(degree(h,x)>0) then
```

```
    h:=resultant(h,f,x): h:=lprem(flist,vlist,h):
  fi: od:
h:
end:

subres:=proc(f,g,x,k)
local i,j,t,r,m,n,a,b,s:
if k=0 then s:=resultant(f,g,x): else
  n:=degree(f,x): m:=degree(g,x): t:=m+n:
  a:=array(1..t,1..t): a:=sylvester(f,g,x):
  r:=m+n-2*k: b:=array(1..r,1..r):
  for j to t do for i to t do
    if (j<=r)and(i<=m-k) then b[i,j]:=a[i,j]:
    elif (j<=r)and(i<=t-k)and(i>m) then
      b[i-k,j]:=a[i,j]: fi: od: od: s:=det(b): fi:
s:
end:

submix:=proc(f,g,x,k,d)
local i,j,t,r,m,n,a,b,s:
if k=0 then s:=resultant(f,g,x): else n:=degree(f,x):
  m:=degree(g,x): t:=m+n: a:=array(1..t,1..t):
  a:=sylvester(f,g,x): r:=m+n-2*k: b:=array(1..r,1..r):
  for j to t do if not(j=t) then for i to t do
    if (j<=r-1)and(i<=m-k) then b[i,j]:=a[i,j]:
    elif (j<=r-1)and(i<=t-k)and(i>m) then
      b[i-k,j]:=a[i,j]: fi: od:
else for i to t do if i<=m-k then b[i,r]:=a[i,t-k-d]:
  elif (i<=t-k)and(i>m) then b[i-k,r]:=a[i,t-k-d] fi:
  od: fi: od: s:=det(b): fi:
s:
end:

uts:=proc(plist,vlist,ulist,g)
local m,n,pu,pulist,h,u,u1,p,i,j,v,r1,x:
m:=nops(ulist): n:=nops(plist): pu:=array(1..n):
h:=g:
for j to m do u:=op(j,ulist): u1:=op(j,ulist0):
```

```
h:=subs(u=u1,h): od:
h:=primpart(h,vlist): r1:=1:
for i from n by-1 to 1 do if not(r1=0) then
  p:=op(i,plist): if not(p=0) then x:=op(i,vlist):
  for j to m do u:=op(j,ulist): u1:=op(j,ulist0):
  p:=subs(u=u1,p): od: if not(p=0) then
  r1:=prem(1,p,x): pu[i]:=primpart(p,vlist): else
  r1:=0: fi: else pu[i]:=0: fi: fi: od:
  if r1=0 then h:=0: else pulist:=convert(pu,list):
    h:=res(pulist,vlist,h): fi:
h:
end:

wdc:=proc(flist,vlist,g)
local f,h,x,hlist,r,s,u,v,t,d,p,q,i,k:
hlist:=flist:  h:=g: v:=0:
for i from nops(flist) by-1 to 1 do
  if v=0 then f:=hlist[i]: x:=vlist[i]: n:=degree(f,x):
    m:=degree(h,x): t:=min(m,n):
    hlist:=subsop(i=0,hlist): r:=0:
    for k from 0 to t do if r=0 then s:=subres(h,f,x,k):
      if not(s=0) then p:=lprem(hlist,vlist,s):
        if not(p=0) then d:=k: r:=1: fi: fi: fi: od:
  if not(r=0) then u:=uset minus convert(vlist,set):
    q:=uts(hlist,vlist,u,p): if q=0 then
    q:=res(hlist,vlist,p): fi: if not(q=0) then
    t:=s*x^d: for k from 0 to d-1 do
      t:=t+submix(h,f,x,d,k)*x^k: od:
    t:=[primpart(t,vlist),i]: v:=1: else h:=p: fi:
  fi: fi: od:
t:
end:
```

索　　引

词条按汉语拼音顺序排列，由特殊符号开始的条目放在最后．

科学家中外译名对照表

Abel N. H.	阿贝尔	Gröbner W.	格罗布纳
Archimedes	阿基米德	Hilbert D.	希尔伯特
Bezout E.	贝佐	Hurwitz W.	赫尔维茨
Bernstein D. N.	伯恩斯坦	Macaulay F. S.	麦考莱
Bourbaki N.	布尔巴基	Mill J. S.	穆勒
Bring	布灵	Minkowski H.	闵可夫斯基
Cayley A.	凯莱	Morley M.	摩勒
Church A.	丘奇	Newton I.	牛顿
Collins G. E.	柯林斯	Pascal B.	帕斯卡
Cramer G.	克莱姆	Routh	路西
Descartes R.	笛卡尔	Simson R.	西摩松
Diophantus	丢番图	Sturm J. Ch. F.	斯图姆
Dixon A. L.	迪克逊	Sylvester J. J.	西尔维斯特
Euclid	欧几里得	Tarski A.	塔尔斯基
Euler L.	欧拉	Taylor	泰勒
Feuerbach K. W.	费尔巴哈	Thébault	泰保
Fourier J.	傅里叶	Tschirnhaus	契尔恩豪斯
Frobenius F. G.	弗罗贝尼乌斯	Turing A. M.	图灵
Gauss G. F.	高斯	Vandermonde A. T.	范德蒙特

参考文献

[1] Arnon, D.S., On mechanical quantifier elimination for elementary algebra and geometry: automatic solution of a non-trivial problem, *Proc. EUROCAL'85,* LNCS **204**, Springer-Verlag, 1985. pp.270-271.

[2] Arnon, D.S., Geometric reasoning with logic and algebra, *Artificial Intelligence,* **37**(1988), 37-60.

[3] Arnon, D.S. & Mignotte, M., On mechanical quantifier elimination for elementary algebra and geometry, *J. Symbolic Computation,* **5**, 237-260 (1988).

[4] Arnon, D.S. & Smith, S., Towards a mechanical solution of the Kahan ellipse problem I, *Proc. EUROCAL'83,* LNCS **162**, Springer-Verlag, 1983. pp.36-44.

[5] Auzinger, W. & Stetter, H.J., An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations, Intern. Series of *Numer. Math.* **86**, 11-30 (1988).

[6] Bernstein, D.N., The number of roots of a system of equations, *Funkt. Anal. Appl.* **9**, 183-185 (1975).

[7] Buchberger, B., Gröbner Bases: an algorithmic method in polynomial ideal theory, *Recent Trends in Multidimensional Systems Theory* (ed. N.K. Bose), D. Reidel Publishing Company, 1985.

[8] Chou, S.C., *Mechanical Geometry Theorem Proving,* D. Reidel Publishing Company, (Amsterdam) 1988.

[9] Chou, S.C., Gao, X.S., Yang, L. & Zhang J.Z., Automated Production of Readable Proofs for Theorems in Non-Euclidean Geometries, WSUCS-94-9, Wichita State University, 1994.

[10] Chou, S.C., Gao, X.S., Yang, L. & Zhang J.Z., A Collection of 90 Mechanically Solved Geometry Problems from Non-Euclidean Geometries, WSUCS-94-10, Wichita State University, 1994.

[11] Chou, S.C., Gao, X.S. & Zhang J.Z., *Machine Proofs in Geometry,* World Scientific Press, 1994.

[12] Collins, G.E. & Hong, H., Partial cylindical algebraic decomposition for quantifier elimination, *J. Symbolic Computation,* **12**, 299-328 (1991).

[13] Dixon, A.L., The eliminant of three quantics in two independent variables, *Proc. London Math. Soc.*, **6**, 468-478, (1908).

[14] Donald, B.R., Kapur, D. & Mundy, J.L., *Symbolic and Numerical Computation for Artificial Intelligence,* Chapter 2, Academic Press, 1992.

[15] Davenport, J.H., Siret, Y. & Tournier, E., it Computer algebra: systems and algorithms for algebraic comutation, Academic Press, 1988.

[16] Fu, H.G., Yang, L. & Zhou, C.C., A geometric approach to Inverse Kinematics, UNU/IIST Report No.**43**, 联合国大学国际软件技术研究所, March 1995.

[17] Feng, G.C., Zhang, S.G. & Liu Y., The structure of solutions to algebraic system and matrices in eigenvalue method, in MM-Preprints, No.**13**, 数学机械化研究中心, 1995. pp.44-55.

[18] Gelfand, I.M., Kapranov, M.M. & Zelevinsky, A.V., it Discriminants, Resultants and Multidimensional Determinants, Birkhäuser, 1994.

[19] Gao, X.S. & Wang, D.K., 'On the automatic derivation of a set of geometric formulae', *Journal of Geometry,* **53**, 79-88 (1995).

[20] Heck, A., *Introduction to Maple,* Springer-Verlag, 1993.

[21] 洪加威，能用例证法来证明几何定理吗？中国科学， 1986 年第 3 期， **234-242**.

[22] 侯晓荣，几何定理机器证明的实验法，学位论文，中国科学院成都计算机应用研究所， 1995 年 5 月.

[23] Huang, Y.Z. & Wu, W.D., A modified version of an algorithm for solving multivariate polynomial system, in MM-Preprints, No.**5**, 数学机械化研究中心， 1990. pp.23-29.

[24] Kapur, D., Saxena, T. & Yang L., Algebraic and geometric reasoning using Dixon resultants, *Proceedings of ISSAC'94*, ACM Press, 1994. pp.99-107.

[25] Lazard, D., Quantifier elimination: optimal solution for two classical examples, *J. Symbolic Computation*, **5**, 261-266 (1988).

[26] Liu, Z.J., Leading coefficients of Sturm sequences and aperiodicity, in MM-Preprints, 数学机械化研究中心， No.**11**, 1994. pp.58-63.

[27] Macaulay, F.S., On some formula in elimination, *Proc. London Math. Soc.*, pp.3-27, May 1902.

[28] Macaulay, F.S., Note on the resultant of a number of polynomials of the same degree, *Proc. London Math. Soc.*, pp.14-21, June 1921.

[29] Mignotte, M., Computer versus paper and pencil, it Proc. CALSIF, **4**, 63-69 (1986).

[30] Mignotte, M., *Mathematics for Computer Algebra*, Springer-Verlag, 1992.

[31] Stillwell, J., Eisenstein's footnote, *The Mathematical Intelligencer*, **17**:2, 58-62 (1995).

[32] Tarski, A., *A Decision Method for Elementary Algebra and Geometry*, University of California Press, Berkeley, 1951.

[33] 吴文俊，初等几何判定问题与机械化证明，中国科学， 1977 年第 6 期， 507-516.

[34] 吴文俊，几何定理机器证明的基本原理，科学出版社，北京， 1984.

[**35**] Wu, W.T., On reducibility problem in mechanical theorem proving of elementary geometries, in MM-Preprints, No.**2**. 数学机械化研究中心, 1987. pp.18-36.

[**36**] Wu, W.T., On problems involving inequalities, in MM-Preprints, No.**7**, 数学机械化研究中心, 1992. pp.1-13.

[**37**] Yang, L., A new method of automated theorem proving, in *The Mathematical Revolution Inspired by Computing*, J.H. Johnson and M.J. Loomes (eds), Oxford University Press 1991. pp.115-126

[**38**] Yang, L & Hou, X.R., Gather-and-Sift: a symbolic method for solving polynomial systems, *Proc. ATCM'95*, Singapore, Dec. 1995.

[**39**] Yang, L., Hou, X.R. and Zeng, Z.B., An alternative algorithm for determining the number of real roots of a polynomial, *International Workshop on Logic and Software Engineering,* Beijing, 1995.

[**40**] 杨路, 侯晓荣, 曾振柄, 多项式的完全判别系统, 中国科学 **E**, (即将发表).

[**41**] Yang, L. & Zhang, J.Z., Searching dependency between algebraic equations: An algorithm applied to automated reasoning, in *Artificial Intelligence in Mathematics*, IMA Conference Proceedings, Oxford University Press, 1994. pp.147-156.

[**42**] Yang Lu, Zhang Jingzhong & Hou Xiaorong, A criterion of dependency between algebraic equations and its applications, *Proceedings of the 1992 International Workshop on Mathematics Mechanization,* Wu and Cheng (eds), International Academic Publishers 1992. pp.110-134.

[**43**] Yang, L., Zhang, J.Z. & Hou, X.R., An efficient decomposition algorithm for geometry theorem proving without factorization, in MM-Preprints, No.**9**, 数学机械化研究中心, 1993. pp.115-131.

[44] Yang L., Zhang J.Z. & Li C.Z., A prover for parallel numerical verification of a class of constructive geometry theorems, *Proceedings of the 1992 International Workshop on Mathematics Mechanization,* Wu and Cheng (eds), International Academic Publishers 1992. pp.244-250.

[45] 张景中，杨路，定理机械化证明的数值并行法及单点例证法原理概述，数学的实践与认识， 1989 年第 1 期， 34-43.

[46] Zhang, J.Z., Yang, L. & Deng, M., The parallel numerical method of mechanical theorem proving, *Theoretical Computer Science* **74**, 253-271 (1990).

[47] 张景中，杨路，高小山，周咸青，几何定理可读证明的自动生成，计算机学报， **18**:5, 380-393, (1995).

[48] 张景中，杨路，侯晓荣，代数方程组相关性的一个判准及其在定理机器证明中的应用，中国科学 **A**, 1993 年 10 期， **1036-1042.**

[49] Yang, L., Zhang, J.Z. & Hou, X.R., A note on Wu Wen-Tsün's nondegenerate condition, *Computer Mathematics,* World Scientific Press, 1993. pp.127-135.

[50] 张景中，杨路，侯晓荣，定理机器证明的结式矩阵法，系统科学与数学， **15**:1, 10-15, (1995).

[51] Zhou, J.P., On the degree of extensions generated by finitely many algebraic numbers, *Journal of Number Theory,* **34**:2, 133-141 (1990).

[52] Liu, Z.J., One general criterion for stability, in MM-Preprints, 数学机械化研究中心， No.**11**, 1994. pp.58-63.

[53] Weispfenning, V., Quantifier elimination for real algebra - the cubic case, *Proceedings of ISSAC'94,* ACM Press, 1994. pp.258-263.

[54] Zhang, J.Z. & Yang, L., A method to overcome the reducibility difficulty in mechanical theorem proving, I.C.T.P. preprint IC/89/263, Trieste, 1989.

[**55**] 张景中，杨路，侯晓荣．定理机器证明的 WE 完全方法，系统科学与数学，**15**:3, 200-207, (1995).

[**56**] Wang, D.M., A decision method for definite polynomial, MM-Preprints, No.**2**, 1987. pp.68-74.